TU Clausthal

Clausthal University of Technology

Mathematical Modelling with Exact, Heuristic and Meta-Heuristic Solution Methodologies for the Fuel-Efficient Platooning of Heavy Duty Vehicles on Road Networks

Abtin Nourmohammadzadeh

Department of Informatics
Clausthal University of Technology

Mathematical Modelling with Exact, Heuristic and Meta-Heuristic Solution Methodologies for the Fuel-Efficient Platooning of Heavy Duty Vehicles on Road Networks

DOCTORAL THESIS
(DISSERTATION)

to be awarded the degree
Doctor rerum naturalium (Dr. rer. nat.)

submitted by
Abtin Nourmohammadzadeh
from Tehran, Iran

approved by the Faculty of
Mathematics/Computer Science and
Mechanical Engineering,
Clausthal University of Technology

Date of oral examination
29th of May 2019

Chairperson of the Board of Examiners
Prof. Dr. Jürgen Dix

Chief Reviewer
Prof. Dr. Sven Hartmann

Reviewer
Prof. Dr. Stefan Westphal

Declaration of Authorship

I, Abtin NOURMOHAMMADZADEH, declare that this thesis titled, "Mathematical Modelling with Exact, Heuristic and Meta-Heuristic Solution Methodologies for the Fuel-Efficient Platooning of Heavy Duty Vehicles on Road Networks" and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a research degree at the Clausthal University of Technology.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

"We should be taught not to wait for inspiration to start a thing. Action always generates inspiration. Inspiration seldom generates action. "

Frank Tibolt

CLAUSTHAL UNIVERSITY OF TECHNOLOGY

Abstract

Faculty of Mathematics/Computer Science and Mechanical Engineering
Department of Informatics

Doctor rerum naturalium (Dr. rer. nat.)

Mathematical Modelling with Exact, Heuristic and Meta-Heuristic Solution Methodologies for the Fuel-Efficient Platooning of Heavy Duty Vehicles on Road Networks

by Abtin NOURMOHAMMADZADEH

As the demand for road transportation steadily increases and simultaneously the fuel price grows, transportation companies face with more challenges to reduce their costs, and more importantly, to mitigate the resulted environmental impacts. One promising approach for the sake of fuel consumption reduction is to make Heavy Duty Vehicles (HDVs) drive together in groups called "platoon" behind each other and in close proximity like a string. This reduces the aerodynamic drag or resistive force on vehicles and as a result less energy or fuel is required for the same movement. Another benefit of platooning is that so the traffic congestion of road networks can be lessened. Although the idea of platooning has been proposed since long ago, its realisation has only become recently possible due to the late advances in information, communications and computation technology. This doctoral thesis proposes a novel and unique approach to Fuel Efficient Platooning (for ease called FEP in the thesis) on large scale road networks. Two different mathematical models are presented which formulate the FEP problem from different aspects. Several real-life attitudes such as travel time constraints, maximum allowable distance, and also multiple speeds for vehicles are embedded in the models, which have not been addressed together in any previous work in the literature.

Consequently, the main contributions, which are our solution methodologies are proposed in three categories. They gradually enable us to tackle bigger sizes of the problem. The first one is attempting the exact solution of the two models by the powerful solver of CPLEX in the GAMS platform. Due to very high computational complexity, some changes are applied to the models to reduce the complexity and increase the biggest solvable size of the problem by the exact solver. The results of the two models and their decomplexified versions are compared together.

Due to the limitations of exact solution approaches in terms of problem scale and the required execution time, heuristic methods are investigated in the next step. Two heuristics from the literature are adapted and modified to deal with our version of the FEP problem. Furthermore, a new efficient heuristic called Global Planning is proposed which is proved to be better than the other two heuristics. Introducing the third group, which are meta-heuristic solution methodologies, is an important part of this thesis. They are designed to deal with samples larger than those solved by the heuristics in a short time. Three famous nature-inspired algorithms which

are also appropriate for our problem, namely: Genetic Algorithm (GA), Ant Colony Optimisation (ACO) and Particle Swarm Optimisation (PSO), are chosen. Then suitable and efficient application procedure for each is devised including specific solution encoding, operators and etc. Subsequently, a thorough comparison is made between the performance of the employed meta-heuristics and heuristics. All comparisons in this thesis are done by an appropriate non-parametric statistical method.

In the final part, some sensitivity analyses are conducted which investigate the effects of changing some important input parameters of the problem.

Generally, this thesis conducts a comprehensive investigation into the FEP problem and provides appropriate solution approaches that can be used in practice for real cases to achieve significant financial and environmental benefits.

Zusammenfassung

Ein vielversprechender Ansatz zur Reduzierung des Kraftstoffverbrauchs von Schwerlastfahrzeugen (HDVs) besteht darin, sie in Gruppen, die als "Platoon" bezeichnet werden, hintereinander und in unmittelbarer Nähe wie eine Schnur zusammenfahren zu lassen. Dies reduziert den Luftwiderstand an Fahrzeugen und führt dazu, dass für die gleiche Bewegung weniger Energie oder Kraftstoff benötigt wird. Diese Doktorarbeit schlägt einen neuen und einzigartigen Ansatz für das Fuel Efficient Platooning (FEP) in großen Straßennetzen vor. Es werden zwei verschiedene mathematische Modelle vorgestellt. Mehrere reale Einstellungen wie Zeitbeschränkungen, maximal zulässige Umwege und auch Mehrfachgeschwindigkeiten für Fahrzeuge sind in die Modelle integriert.

Folglich werden die wichtigsten Beiträge, die unsere Lösungsmethoden sind, in drei Kategorien vorgestellt. Die erste versucht, exakte Lösungen für die beiden Modelle durch den leistungsstarken Solver von CPLEX in der GAMS-Plattform zu finden. Aufgrund der sehr hohen Rechenkomplexität werden einige Änderungen in den Modellen vorgenommen, um die Komplexität zu reduzieren und die größte mit dem exakten Solver lösbare Größe des Problems zu erhöhen. Die Ergebnisse der beiden Modelle und ihre dekomplexisierten Versionen werden miteinander verglichen.

Im Hinblick auf die Grenzen exakter Lösungsansätze bezüglich der Problemgröße und der erforderlichen Ausführungszeit werden im nächsten Schritt heuristische Methoden vorgestellt. Zwei Heuristiken aus der Literatur werden angepasst und modifiziert, um unsere Version des FEP-Problems zu behandeln. Darüber hinaus wird auch eine neue und effiziente Heuristik namens Global Planning vorgeschlagen, die sich als besser erweist als die beiden anderen Heuristiken.

Die Einführung der dritten Gruppe, die unsere meta-heuristischen Lösungsmethoden beinhaltet, ist ein wichtiger Teil dieser Arbeit. Diese sind so entworfen, dass sie mit Stichproben größer als die mit den Heuristiken gelösten umgehen können. Drei berühmte von der Natur inspirierte Algorithmen, die auch für unser Problem geeignet sind, nämlich: Genetischer Algorithmus (GA), Ameisenkolonieoptimierung (ACO) und Partikelschwarmoptimierung (PSO) sind ausgewählt. Anschließend wird ein geeignetes und effizientes Applikationsverfahren für jede einzelne Meta-Heuritik entwickelt, einschließlich spezifischer Lösungscodierung, Operatoren usw. Folglich wird ein gründlicher Vergleich zwischen der Leistung der verwendeten Meta-Heuristiken und der Heuristiken durchgeführt. Alle Vergleiche in dieser Arbeit werden mit einer geeigneten nicht-parametrischen statistischen Methode durchgeführt.

Im letzten Teil sind einige Sensitivitätsanalysen durchgeführt, die die Auswirkungen der Änderung einiger wichtiger Eingabeparameter des Problems untersuchen.

Im Allgemeinen führt diese Arbeit eine umfassende Untersuchung des FEP-Problems durch und liefert geeignete Lösungsansätze, die in der Praxis für reale Fälle verwendet werden können, um erhebliche finanzielle und ökologische Vorteile zu erzielen.

Acknowledgements

I would like to give my special thanks to my supervisor, Prof. Sven Hartmann, and express my sincere gratitude to his valuable helps, advices and guidance during my doctoral program at the Clausthal University of Technology.

In addition, I would like to use this chance to thank my parents for all their continuous and endless supports and encouragement, which have been the source of any success throughout my life.

Contents

List of Abbreviations

ACO	**A**nt **C**olony **O**ptimisation
AHS	**A**utomated **H**ighway **S**ystems
B&C	**B**ranch and **C**ut
BL	**B**enefit **L**oss
BP	**B**est **P**air (heuristic)
CACC	**C**ooperative **A**daptive **C**ruise **C**ontrol
DCC	**D**ecentralised **C**ongestion **C**ontrol
DOE	**D**esign of **E**xperiments
DSRC	**D**edicated **S**hort **R**ange **C**ommunication
DynB	**D**ynamic **B**eaconing
ESTI	European Telecommunications Standard Institute (ETSI)
FEP	**F**uel **E**fficient **P**latooning
FMS	**F**leet **M**anagement **S**ystem
GA	**G**enetic **A**lgorithm
GAMS	**G**eneral **A**lgebraic **M**odelling **S**ystem
GP	**G**lobal **P**lanning (heuristic)
GPS	**G**lobal **P**ositioning **S**ystem
H	**H**ub (heuristic)
HDV	**H**eavy **D**uty **V**ehicle means the same as Truck in this thesis
HEV	**H**ybrid **E**lectric **V**ehicle
ITF	**I**nternational **T**ransport **F**ederation
ITS	**I**ntelligent **T**ransport **S**ystem
LS	**L**ocal **S**earch (heuristic)
MPC	**M**odel **P**redictive **C**ontrol
PRT	**P**ersonal **R**apid **T**ransit
RSM	**R**esponse **S**urface **M**ethod
PSO	**P**article **S**warm **O**ptimisation
RV	**R**est **V**iolation
SSPP	**S**ame **S**tart **P**latooning **P**roblem
SUMO	**S**imulation for **U**rban **Mo**bility
VRP	**V**ehicle **R**outing **P**roblem
WBCSD	**W**orld **B**usiness **C**ouncil for **S**ustainable **D**evelopment

Dedicated to my dear mother and father

Chapter 1

Introduction

The world is experiencing an ever increasing demand for freight transportation, which has resulted in more fuel consumption, and more importantly, in rising environmental impacts by greenhouse gas emissions. Hence, transportation companies seek novel and practical approaches to reduce the fuel consumption of their Heavy Duty Vehicles (HDVs) even for a few per cent . Platooning, which means driving vehicles behind each other in close proximity in a lane, has recently caught attention as a promising strategy to decrease the fuel usage of HDVs and achieve some other important benefits. This method works as driving in the vicinity of other vehicles can reduce the aerodynamic drag or resistive force, thus, less energy is required for the movement. Fuel Efficient Platooning (FEP) is a very complex problem. Hence, in this thesis, we invest in deriving appropriate models and solution methodologies to deal with it. This chapter aims at shedding light on the investigated FEP, as well as the motivations and contributions of this thesis.

The chapter is organised as follows: in Section 1.1, the motivations and necessity of the subject, and also its difficulties are discussed. Section 1.2 addresses the contributions of this thesis. An outline of the thesis is presented in Section 1.3. Finally, the summary of this chapter is given in Section 1.4.

1.1 Motivations and Problem Description

The worldwide growth in production and trade has led to larger transportation volumes on roads by HDVs (trucks). This causes a major increase in fuel combustion, which incurs huge expense to transportation companies because as it is shown in Figure 1.1, the fuel cost constitutes almost 30% of the life-cycle operational cost of a truck in Europe (*Scania CV AB (2012)* [1]). By growing the oil price, this percentage will become even larger. Furthermore, vehicles comprise a considerable 20% of the total carbon emissions of which a quarter comes from HDVs (*Schroten et al., 2012* [80]). Therefore, even a minor reduction in the fuel consumption provides considerable financial and environmental benefits.

The World Business Council for Sustainable Development (WBCSD) predicts an annual growth rate of 2.4% for tonne-kilometer transported goods of heavy-duty vehicles (HDVs) in the period 2000–2050 (*WBCSD* [88], [59]). This will result in a total increase of over 200% in tonne-kilometre transported goods in the end. A later forecast published by International Transport Federation (ITF) in 2015 also shows that the worldwide freight volumes transported on road, which was 6388 billion tonne-kilometre in 2010, will be almost 31000 billion in 2050 [92]. In such a situation, the

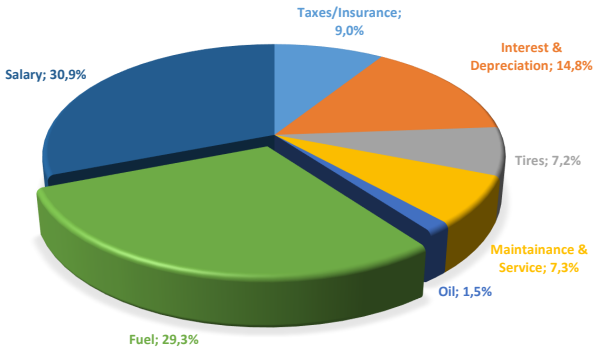


FIGURE 1.1: The share of different costs in the total life-cycle cost of European fleet owners (*Scania CV AB, 2012* [1]), which is similar to the life-cycle costs of European HDVs (*Schittler, 2003* [78]).

European Commission has set goals to achieve a more competitive and resource-efficient transport system. The key goal is to decrease the emissions in the transport sector by 60% by 2050 in order to reduce the environmental impacts to avert climate change and maintain a sustainable environment (*European Commission, 2011* [16]). Simultaneously, the growing traffic intensity has made the roads saturated throughout the major cities in the world on almost every weekday in the morning and evening (*Alam, 2014* [3]).

Both HDV manufacturers and fleet owners are confronted with critical challenges to maintain a sustainable environment. They have to follow legislations and policies as well as making the transport more fuel efficient due to the rise of fuel prices. Hence, they are searching for novel applicable and efficient methods to cut their fuel consumption (*Larsson et al. 2015* [54]).

The use of Fleet Management Systems (FMSs) is increasing among fleet operators. The FMS enables the fleet operator to analyse and monitor the operation and condition of each vehicle, for instance, the amount of coasting, idling, braking, fuel consumption, speed, and position. As the position of vehicles is very important in FMSs, a Global Positioning System (GPS) device is required in the vehicles (*Liang, 2014* [59]). Using FMSs allows the transportation companies to implement a practical fuel saving approach called platooning. In this method, strings of HDVs (for ease also generally called vehicles or trucks throughout this thesis) are formed in which HDVs drive close behind each other to reduce the aerodynamic drag or resistive force as it is shown in Figure 1.2. Since the following vehicles drive in the slipstream of another vehicle which drives ahead, the normal fuel consumption of every trailing truck can be reduced based on a saving factor. Figure 1.3 depicts a schematic view of a platoon. The saving factor of platooning depends mainly on the speed of vehicles (s) and the distances between them (d). According to *Robinson et al. (2010)* [77], fuel reductions of up to 20% are achievable for the non-leading vehicles. As *Bonnet and Fritz (2000)* [12] show, if HDVs travel at 80 km/h and the distance between them is 10m, a fuel reduction of 21% can be achieved by the trailing vehicles, while the fuel reduction is 16% for an inter-vehicle distance of 16m. Obviously, safety issues must be considered by forming a platoon because vehicles are driving at such close distances from each other. This has been the focus of the majority of platooning works.

FIGURE 1.2: An Example of a Platoon in reality with four trucks driving in close distance behind each other (The picture is from the website of Ontario Trucking Assassination, http://ontruck.org/)

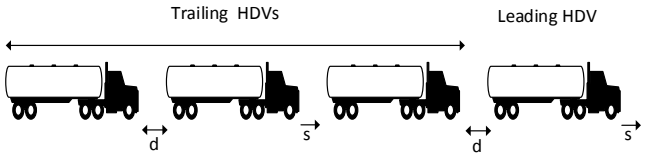


FIGURE 1.3: Schematic view of a platoon

In platoons, vehicles drive cooperatively like migration of birds or a group of dolphins. The formation of birds in the migration is aerodynamically efficient, and dolphins swim without collision while communicating with each other (*Kavathekar and Chen, 2011* [46]).

This cooperative driving can, furthermore, contribute to a major reduction in large wastes of traffic congestion and provide an increase in the road capacity. This is because of shorter headway distances, which makes HDVs gather close to each other. Hence, the total road capacity can be increased. This is very crucial when we, for example, observe the related studies like the one conducted by *Schrank et al. (2012)* [79]. It shows that road congestion annually wastes 5.5 million man-hours and 2.9 billion gallons of fuel, which cost collectively 121 billion dollars. This will definitely be worsened by growing of urban population. An explicit example of the road congestion is when a typical highway which can serve 2200 vehicles per hour is occupied by only 18% of this capacity due to large inter-vehicular distances (*Highway Capacity Manual, 2010* [38]).

It is proved that those vehicles equipped with Cooperative Adaptive Cruise Control (CACC) which are able to drive in platoons improve traffic flow and use the road space more efficiently (*Lu and Shladover, 2014* [64]; *Nowakowski et al. 2010* [71]). According to some related simulation studies (*Nowakowski et al., 2010* [71]; *Shladover, 2012* [83]; *Lioris et. al. 2016* [62]), CACC increases road capacity even further from 2200 vehicles per hour to almost 4000 vehicles per hour if it is used by the whole vehicles.

Additionally, governing vehicle platoons by an automated control strategy, the overall traffic flow and safety is expected to improve (*Kavathekar and Chen, 2011* [46]). An appropriately managed platoon can potentially offer

enhanced safety, improved highway utility, increased fuel economy, and reduced emissions (*Xu et al. 2014* [96]). Moreover, if there is the possibility to drive the trucks of platoons only by a driver in the leading truck, another significant amount of saving can be achieved because according to Figure 1.1, the drivers' salary corresponds to the largest proportion in the total cost. However, this requires special technology and tools which have not become popular yet. Therefore, this case is not considered in our present work and we only investigate the platoon formation with the goal of reducing the fuel consumption, i.e. FEP problem. Some pre-requirements of autonomous driving are addressed in *Kühn et al. (2017)* [49].

Platoons do not take shape spontaneously on road networks, in other words, we cannot simply send vehicles on their way hoping to make platoons. Given a set of vehicles with their origins, destinations, time restrictions, the fuel saving factor of platooning, speed options, maximum allowable detour and also the topology of the road network as the inputs, we need to determine a fuel efficient route and travelling time schedule for each vehicle as the outputs of this version of the FEP problem. Hence, platooning can also be defined as a Vehicle Routing Problem (VRP) which concerns minimizing the fuel consumption by making groups of vehicles to drive together. If it is beneficial, some vehicles should deviate from their shortest path and take a detour in order to join a platoon. For some vehicles, it could be more worthwhile to increase their speed temporarily to reach a platoon at some time in the future, whereas for others it might be more efficient to either stop and wait or keep the current velocity and travel alone.

According to *Kammer (2013)* [43] and *Varaiya (1993)* [90], the entire platooning problem consists of three layers shown by Figure 1.4. Although they are very different in nature, they require all a full application of the platooning concept. Starting from the bottom, layer 3 or platoon controlling focuses on the actual control of individual vehicles and platoons. It deals with issues like inter-vehicle communication, string stability and individual speed profiles. In the road planning level, the velocity of platoons is determined based on the road profile and vehicles' data. Finally, the first layer coordinates multiple vehicles and platoons over large networks, optimises routes, and schedules the formation and splitting of platoons on a general level.

The problem addressed in this thesis, belongs to the first layer and a little bit second. This means that a global view of the road network is considered, involving many vehicles and platoons at once rather than concerning the actual behaviour of vehicles.

In our FEP, we have a road network graph with a number of nodes representing cities or locations and some connections or edges between the nodes representing roads, a set of vehicles (trucks) with their origin, destination, earliest possible departure time from the origin (release time) and latest allowable arrival time at the destination (deadline). In addition, for any vehicle, a maximum extra distance over the shortest path length is considered which must not be exceeded. There are also a set of allowable speeds for vehicles which can be used for accelerating or decelerating to help them joining platoons. There is a fuel saving factor (rate) of platooning for every following vehicle which differs for each speed. Higher the speed is, larger is the saving factor. On the other hand, fuel usage per unit of distance increases by higher speeds. The expected outputs are a routing and travel

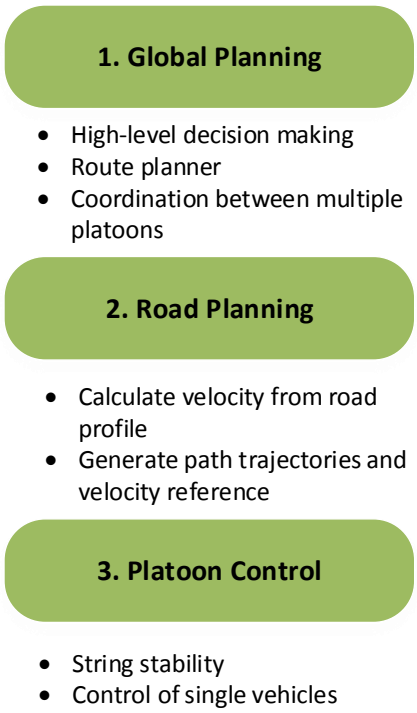


FIGURE 1.4: Decision layers of platooning presented by *Kammer (2013)* [43] and *Varaiya (1993)* [90]

schedule for each vehicle which result together in the minimised total fuel consumption (cost) corresponding to all vehicles or the whole system. In the next section, the goals of this thesis are explained.

1.2 Contributions

Most of the previous researches into HDV platooning have not dealt with the question: how should platoons be formed? They have involved with the maintenance and safe manoeuvring of already existing platoons. Only few number of works have considered the absolutely important problem of platoon formation to reduce the fuel consumption, thus, there is a strong need for more attempts in this direction.

The first presentation of platooning idea was by *Levine and Athans* [56] in 1966, however, its application for real life cases has been postponed for long and become only possible in recent years due to its complexity, lack of necessary equipments and appropriate infrastructure. Despite the huge improvements in computing power in comparison to the past, solving an FEP problem in real sizes and conditions is still very difficult due to the high computational complexity and very long required solution time. Hence, this thesis attempts to introduce practical approaches which are able to tackle a comprehensive version of the FEP problem.

This thesis formulates the Fuel Efficient Platooning (FEP) by presenting two mixed-integer linear mathematical models. One counts the platooning effects based on the same chosen edges at the same time, while in the other model the platoons are noticed wherever any vehicle directly follows another one. These models try to comprise more attitudes which emerge in real life

cases, and have not been investigated and taken altogether into account in the previous literature.

These attitudes are time constraints on the earliest departure, latest possible arrival time and a maximum allowable detour distance of vehicles, and also the existence of a set of allowable speeds for each vehicle.

The time constraint for the vehicles' departure is referred to as the release time in scheduling problems. It is according to the fact that we cannot make a vehicle leave its origin before a specific time because its load is not ready or even the driver or vehicle is not available. On the other hand, it is not possible to delay the arrival of the vehicle or delivery of the load at the destination to after a specific time because, for example, there are foods or things which may decay or parts which must be delivered to factories on time for production, and similar reasons.

In addition to deadlines, there is a distance related constraint that does not allow vehicles to choose paths which are longer than a maximum length. This is because drivers do not choose such long routes in reality.

The assumption of various speeds is very natural because vehicles do not drive on their whole way at the same speed. For platooning, they may accelerate to join other vehicles which are further away or slow down for some others to arrive and join.

Three of the sample road network graphs used in this work are based on the real areas namely: Chicago, Germany and Sweden. Furthermore, the FEP problem is examined on some hard grid shape networks of three different sizes. Our experiments include various numbers of vehicles from small to large with an enough number of runs for each. Our vehicles have random origin, destination, earliest starting time, and deadlines.

The small instances are tackled by an exact solver based on the two models. Some strategies are tried to decomplexify the models and increase the biggest tractable size by the solver. Comparing the models and their decomplexified versions is a specific approach in this thesis.

However, the FEP problem is so complex that other alternative solution approaches are in demand to deal with larger sizes. Hence, in the next step, some practical heuristic algorithms are given. The solving process is based on two phases: firstly routing, and then scheduling with the aid of some embedded algorithms, which are within our important contributions. This approach is very unique. Three heuristics are examined. In two of them, some concepts presented in the previous works are used, whereas one is completely original of this work.

Since tackling our problem in real size requires faster solution approaches, three famous and applicable nature-inspired meta-heuristcs are adapted or redesigned and employed. They are Genetic Algorithm (GA), Ant Colony Optimisation (ACO) and Particle Swarm Optimsiation (PSO). Suitable solution encoding and adjustments are considered for each algorithm to become applicable to our FEP problem. It can be stated that the meta-heuristics are the main contributions or heart of this thesis because to the best of our knowledge there has not been any considerable attempt at applying such methods to the FEP problem.

Generally, this thesis tries to model the FEP problem with more details and proposes various appropriate solution methodologies to deal with it in large sizes which usually arise in reality. A thorough comparison is made between the performance of the applied approaches by proper statistical

tests. In summary, the main contributions of this thesis can be listed as below:

- Proposing holistic mathematical models for the FEP problem which can better represent the real situation by considering some concepts like release times, deadlines, maximum allowable detours and different alternative speeds for vehicles.
- Generating various test problems on different graphs which are partly based on the situation on real road networks.
- Obtaining exact solutions for the smaller samples by an appropriate exact solver.
- Reducing the complexity of the problem for the exact solver by the aid of some effective decomplexifying methods.
- Developing heuristic methods based on some available and also new concepts .
- Providing appropriate algorithms to derive an excellent time scheduling and speed adjustment from a routing solution.
- Using a variety of applicable meta-heuristics and evolutionary optimisation algorithms by devising suitable solution encoding schemes and operators to solve large-scaled instances of the problem.
- Having thorough comparisons among the proposed methods by appropriate non-parametric statistical tests.
- Conducting sensitivity analyses which investigate the effect of changing some inputs such as the number of vehicles, fuel saving factor, softening the time constraints and etc.
- Analysing the problem from the perspective of an individual participant or vehicle.

At the end of this section, a graphical view of the research agenda or main contributions of this thesis is illustrated in Figure 1.5.

1.3 Thesis Outline

This thesis is organised in altogether 8 chapters. In this chapter, the underlying problem as well as its motivations, scopes and contributions were explained. Chapter 2 covers the literature review and background of the whole field of vehicle platooning. Chapter 3 contains our proposed mathematical formulations and the related descriptions. It follows with the explanations about the used road network graphs and creating the test problems. Subsequently, the exact solutions of the models, the complexity reduction strategies and their effects are presented in Chapter 4. The heuristic methods and their results are introduced in Chapter 5. Chapter 6 explains our three meta-heuristic approaches and continues with the presentation and comparison of their results. Chapter 7 comprises important sensitivity analyses with changing the amount of some inputs, constraints and settings. Finally, in Chapter 8, the conclusions of this thesis are drawn and some directions for future research are suggested.

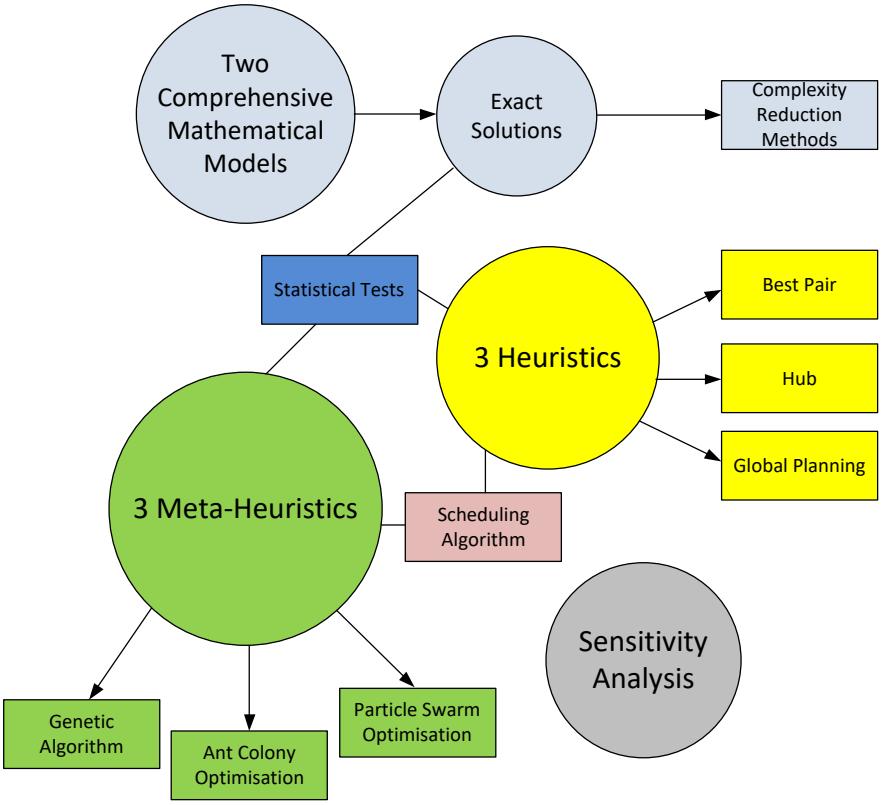


FIGURE 1.5: The research agenda of this thesis

1.4 Summary

In this chapter, the focused FEP problem and the necessity of investigation into it were explained. Then, our main contributions and goals were introduced. Finally, the organisation and outline of this thesis were described. The next step is to give an overview of the main works done in the platooning field, which is the subject of the following chapter.

Chapter 2

Background

In this chapter, an overview of the conducted researches into vehicle platooning is provided. So far most of the works have not addressed the formation of platoons and they have mainly dealt with safety and technological issues. The reviewed works are classified and presented in two parts: in Section 2.1, platooning works whose focus is not on the fuel-efficient platoon formation are shortly introduced. They mainly investigate developing new platooning technologies to accomplish safe manoeuvring of HDVs. Subsequently, Section 2.2 introduces fuel efficient platooning (FEP) works, which are very related to the subject of this thesis. Generally, this chapter aims at presenting a comprehensive literature review of the whole field of platooning, however, it does not reflect all the existing attempts. In the end, a summary of this chapter is presented in Section 2.3.

2.1 Technologies, Controlling and Safe Manoeuvring

In this section, some previous attempts in the general field of platooning are reviewed which have mainly focused on maintenance and safe manoeuvring of trucks (vehicles) within the given platoons. Although these surveys are not very related to our research, the concepts and principles of HDV platooning especially the technical information that they provide are very useful in the area of FEP. Moreover, without being ensured about the safety, we cannot go further and think of building any platoon.

One of the pioneers in presenting the idea of cooperation between vehicles is *General Motors (GM)* at the New York World's Fair in 1939. Their visioning film entitled "New Horizons" [87] presents an image that cars would drive autonomously with the help of curved sides to keep them in the lane, and with automatic radio control to maintain safe distances between the cars at unreduced speed (*Liang, 2014* [59]).

In *Kavathekar and Chen (2011)* [46], the vehicle platooning literature published between 1994 and 2010 is categorized and discussed. The paper includes a general introduction and overview on vehicle platooning and a technical description of the methodology. *Davila and Nombela (2012)* [20] provide an overview of the benefits of platooning systems in order to improve awareness and acceptance of platoons by the general public and relevant stakeholders.

Alam et al. (2015) [4] present a linear quadratic control framework for the design of a high-level cooperative platooning controller suitable for modern HDVs. They utilise a non-linear low-level dynamical model, where realistic response delays in certain modes of operation are considered. The controller performance is evaluated by numerical and experimental studies. *Al-kaisy*

et al. (2011) [2] do an empirical investigation into platooning on two-lane two-way highways. They use six data sets from three study sites in the state of Montana in the USA.

A part of platooning research has focused on designing appropriate communication systems for platooning. A developed vehicle-to-vehicle (V2V) communication system which enables vehicles to drive in platoons is explained by *Bergenhem et al. (2012)* [7]. This system allows forwarding of messages between vehicles to share data such as vehicle speed. They present performance measurements of a first prototype of the V2V system. *Karlsson et al. (2016)* [44] perform V2V communication field tests in a platoon consisting of four trucks where it was assumed that large vehicles such as trucks need multiple antennas to overcome shadowing and obstruction caused by the vehicle itself, trailers and other trucks in the platoon. *Larsson et al. (2016)* [55] propose a broadcast message forwarding algorithm for V2V communication. Their algorithm is evaluated by simulation using real world V2V measurement data as input. In the paper of *Zhao et al. (2015)* [101], an embedded communication system is developed for small automatic electric vehicle. An algorithm is implemented to avoid transmit collision in the developed system. Based on the V2V communication, a platooning control system including vehicle interruptions and separations is designed.

Segata et al. (2015) [81] investigate different communication strategies by explicitly taking into account the requirements of the controller, exploiting synchronised communication slots, and transmitting power adaptation. They compare the proposed approaches to two state-of-the-art adaptive beaconing protocols that have been designed for cooperative awareness applications. They are the European Telecommunications Standards Institute (ETSI), Decentralized Congestion Control (DCC) and Dynamic Beaconing (DynB). *Bom et al. (2005)* [11] address platooning of automatic guided vehicles, relying on RTK-GPS sensors and inter-vehicles communication. *Gao et al. (2016)* [32] studies Dedicated Short Range Communication (DSRC) Vehicle-to-Vehicle (V2V) performance in truck platooning scenarios through real-world experiments. Intra-platoon information management strategies for dealing with safe and stable operation are proposed in *Fernandes and Nunes (2012)* [30]. New algorithms to mitigate communication delays are presented, and Matlab/Simulink based simulation results are reported.

Some works have been done in developing control systems for platooning. *Kianfar et al. (2015)* [50] presents a predictive control strategy for vehicle platoons, accommodating both string stability and constraints (e.g., physical and safety) satisfaction. *Hoang et al. (2015)* [39] derive an efficient messaging scheme based on relay selection which minimises the probability of error at the intended receiver(s) for both unicast and broadcast to disseminate messages within a platoon. *Davis (2013)* analyses the dynamics of a platoon of adaptive cruise control vehicles for a general mechanical response and tests the effects of acceleration-feedback control. He presents a methodology to calculate the dynamic response of a platoon of adaptive cruise control vehicles given the mechanical response function of an individual car. *Linsenmayer et al. (2015)* [61] investigate the control of a platoon with a non-linear controller under event-triggered communication and *Wang et al. (2005)* [91] studies the motion and control of a vehicle platooning system by providing a co-simulation platform. In the latter, a centralized linear quadratic regulator

system for controlling the vehicles in the platoon has been developed considering the aerodynamic characteristics of the vehicle and the resistance due to the road slope. *Di Bernardo et al. (2016)* [23] presents a novel control design framework for vehicle platooning together with its experimental validation. They formulate the problem of controlling the vehicles within a platoon as a high-order network consensus problem considering their desired velocities and inter-vehicle distances. *Masehuw et al. (2008)* [66] presents two control designs for heavy-duty truck platoons with constant spacings that focus on improved safety margins in longitudinal and lateral vehicle following.

Not dealing with any platoon formation, *Kaku et al. (2012)* [42] presents an ecological vehicle platooning control system that aims at reducing overall fuel consumption of the vehicles in a platoon. Similarly, *Turri et al. (2016)* [89] propose a two-layer control architecture for heavy-duty vehicle platooning aimed at safely and fuel-efficiently coordinating the vehicles in the platoon. Dynamic programming is used to compute the fuel-optimal speed profile for the entire platoon and a distributed model predictive control framework is developed for the real-time control of vehicles.

The paper of *Zheng et al. (2016)* [102] studies the scalability limitations of large-scale vehicular platoons moving in rigid formation, and proposes two basic ways to improve stability margins, i.e., enlarging information topology and employing asymmetric control. *Deng (2014)* [21] focuses on quantification of the impacts of HDV platooning on traffic flow and initialises a simulation framework. In the research of *Sungu et al. (2015)* [25], a new non-linear spacing policy and a linear feedback controller related to the policy are proposed to achieve string and traffic flow stability.

Some platooning models have been proposed in this category of works. *Paoletti and Innocenti* (2015) [73] introduce a novel platooning model for vehicles subject to non-linear drag. A linear parametric controller, taking into account both the actual position and the relative displacement with respect to neighbours, is applied to each vehicle. Then its effects are investigated via a semi-analytical approach based on partial differential equations. *Liag et al. (2015)* [60] study how traffic may affect a merging of two HDVs trying to form a platoon by simulating this for different traffic densities and HDV speeds. *Fernandes and Nunes (2010)* [29] study new models to allow the research of cooperative and autonomous communication-enabled vehicles, with platooning capabilities, and the addition of new features in Simulation for Urban Mobility (SUMO).

In the paper of *Ogitsu et al. (2012)* [72] , they propose a decision process for handling operations against assumed system failures in platooning with the aim of applying the process to their platooning system developed in Energy-Saving ITS Technologies Project. *Goli & Eskandarian (2014)* [34] study the problem of vehicle platooning with a particular focus on evaluation of lateral trajectories when one or several vehicle(s) merge(s) from the adjacent lane into the main vehicle platoon under longitudinal control. *Deng et al. (2014)* [21] attempts to investigate the energy saving potential of truck platoons by intelligent speed planning. *Guo et al. (2012)* [35] introduce a self-defensive coordinated manoeuvring strategy to generalise platooning to situations with non-automated interfering vehicles in mixed traffic.

Michaud et al. (2006) [67] formulates the problem in terms of sensing and communicated information. By emulating platoons using a group of mobile robots, the authors demonstrate the feasibility of manoeuvres (such

as entering, exiting, and recuperating from an accident) using different distributed coordination strategies. *Espinosa et al. (2007)* [27] present a method for platoon formation and control in linear and non-linear trajectories. *Lu and Hedrick (2000)* [63] propose a mathematical model and a new concept virtual platooning with an adaptive solution algorithm. The paper of *de Wit et al. (1999)* [94] reviews some of up to that time existing stability definitions (i.e. string stability), and discusses how the available platoon information and separation policies influence the stability results.

Van de Hoef et al. (2016) [40] propose a centralized system that provides trucks with routes and speed profiles allowing them to dynamically form platoons during their journeys. *Li (2017)* [57] presents a two-part paper. The first part develops a vehicle platoon model to capture the dynamics of vehicles grouping behaviour and proposes an online platoon recognition algorithm, while the second part investigates various important characteristics of vehicle platoons and derives their statistical distribution models, including platoon size, within-platoon headway, between-platoon headway and platoon speed. In another paper [58], *Li* employs a Markov regime-switching stochastic process to model the dynamic behaviour of platoon-to-platoon transitions, and a state space model which is employed to describe individual vehicles dynamic movements within each vehicle platoon.

The work of *Xu et al. (2014)* [96] focuses on quantitative characterisation of the impact of communication information structures and contents on platoon safety. They did a quite similar research in *Xu et al. (2013)* [97].

Meanwhile, few optimisation works can be observed such as *Van Willigen et al. (2013)* [93], which proposes a multi-objective evolutionary algorithm based on NEAT and SPEA2. It evolves high level controllers which guide vehicles to join and leave platoons. The algorithm yields a set of solutions that each embody their own prioritisation of various user requirements such as speed, comfort or fuel economy. Hence, it can also belong to the works of the next section but since its contribution to controlling is dominant, it is presented here.

Some works addressed the safe platooning of vehicles by investigating a local paradigm. *Dafflon et al. (2013)* [19] studies side by side vehicle platoon systems. They present a local control approach with obstacle avoidance. The proposed vehicle decision-making process works as a multi-agent system, in which the agents make collectively the best decision according to the perceived constraints and the preceding vehicle position as well as the speed. *Zaher et al. (2016)* [26] do column platoon configurations with two main goals. Firstly, to propose a local approach of platooning where each vehicle is considered to be independent and able to decide about its references such as speed and orientation in a local frame based only on its perceptions. The second goal is to show the effect of the inertial force of the local frame. To evaluate the importance of this inertial force, a comparison is made between the behaviour of the vehicle in case that once the inertial force is considered and once it is not.

Some works investigate platooning with the assumption of some modern technologies and infrastructures. In *Alvarez & Horowitz (1999a)* [5], the problem of designing safe controllers for vehicle manoeuvring in Automated Highway Systems (AHS) is addressed where traffic is organised into platoons of closely spaced vehicles. Conditions to achieve safe platooning under normal modes of operation are investigated. In their other work,

Alvarez & Horowitz (1999b) [6], the design of a velocity tracking controller for safe vehicle manoeuvring is presented. The notion of safety is related to the absence of collisions that exceed a given relative velocity threshold. *Gattis et al. (1997)* [33] develop a simple linear regression equation to express the proportion of vehicles in platoons as a function of the total one-direction traffic volume. *Parent et al. (1996)* [74] gives a vision technique for platoon driving of a fleet of homogeneous electric cars. *Featherstone & Lowson (2009)* [28] consider the practical safety implications of platooning for both Automated Highway Systems (AHS) and Personal Rapid Transit (PRT) systems.

2.2 Fuel Efficient Platooning

As mentioned, only a comparatively small number of researches have investigated the FEP. A good review of such works with some significant categorisation is available in *Bhoopalam et al. (2017)* [9].

Liang (2014) [59] introduces a framework for the HDV platooning based on analysing and validating the possibility to form platoons through fuel-efficient coordination decisions consisting of re-routing, adjusting departure times, and adjusting speed profiles. A functional architecture for goods transport is presented which divides the overall complex transport system into manageable layers. The impact of a coordination decision on the fuel is computed by a developed vehicle model. The focus is on adjusting vehicles' speeds through catch-up coordination.

Larsson et al. (2015) [54] propose an integer linear programming model for the problem where deadlines of trucks are not taken into account. They prove that the problem is an NP-hard one and very time consuming to be solved in large scales even in the case that the road network graph is planar. Two heuristics, namely best pair and hub, with a local search are applied to the problem and their performance are compared with the optimal solutions on the German Autobahn road network. Their results show that with a fuel reduction factor of 10% by platooning, fuel savings of 1–2% for as few as 10 trucks are achieved and the saving percentage increases with the number of trucks. If all trucks (vehicles) start at the same point, savings of up to 9% are obtained for only 200 trucks.

Yu et al. (2016a) [98] presents a new model predictive control system (MPC) for hybrid electric vehicle platooning (HEV) using route information to improve fuel economy. First, a system for hybrid electric vehicle platooning is developed considering various speeds. Second, a general model of road slope in a vehicle platoon is developed. Third, the effect of prediction horizon and sampling interval on battery efficiency is analysed. The model predictive control problem is solved using a discrete numerical computation method, the continuation and generalised minimum residual method. Computer simulation results reveal improvements in fuel economy using the proposed control method. In another research, namely *Yu et al. (2016b)* [99], they use slope information to create a similar MPC for HEV platooning.

An experimental study on reducing aerodynamic drag and improving fuel economy through vehicle platooning was conducted in *Tadakuma et al. (2016)* [86] to develop an Intelligent Transport System (ITS) with good fuel economy of the entire vehicle-based transportation society. The objectives of the study are to achieve a simple and quick approach to estimating

the aerodynamic drag reduction rates of vehicle platooning. *Koller et al. (2015)* [51] focuses on the optimal control of merging manoeuvres for the formation of a growing platoon. The merging problem is formulated as a hybrid optimal control problem and an algorithm for the computation of optimal merging times and corresponding optimal vehicle trajectories is developed by exploiting an extension of Pontryagin's maximum principle. In addition, they present a model predictive control approach on the basis of this algorithm which makes the merging manoeuvres robust to modelling uncertainties and external disturbances.

Kammer (2013) [43] presents a comprehensive mathematical formulation and a global solution approach for the problem in his thesis. A locally distributed approach is applied to the problem due to an exponential increase in its computational complexity. Moreover, he investigates the influence of the number of trucks on the total fuel savings and examines the effect of limiting the maximum travel time of trucks.

Hoef et al. (2015) [41] tackle the fuel-optimal coordination of trucks into platoons by medoids clustering and study how fuel-optimal speed profiles for platooning can be computed. A first-order fuel model is considered and pairwise optimal plans are derived. They formulate an optimisation problem that combines these pairwise plans into an overall plan for a large number of trucks, then, propose an approximation algorithm similar to the partitioning around medoids algorithm and discuss its convergence. *Steinmetz et al. (2017)* [85] propose a novel spatial grouping approach to determine near-optimal groups for platooning. Their method uses fast geometrical heuristics, consisting of direction-comparison, a modified version of a geometric-median-calculation, and a comparison of intersections areas between two vehicles.

Larson et al. (2016) [52] address both the issues of the platoon coordination and vehicle routing in a novel linear mathematical model. Then they solve the model in GAMS [1] language with Gurobi [2] solver for instances which are lager than those of previous works.

In the article, *Nourmohammadzadeh and Hartmann (2016)* [70], we present a model for the platooning problem, and solve instances containing 10 to 50 trucks on a network inspired from locations of 20 important cities in Germany. Then a Genetic Algorithm with a novel solution encoding system is presented and its results are compared with the optima. This verifies the goodness of the solution methodology for instances in such scales. Continuing our research on designing meta-heuristics for the FEP problem, in *Nourmohammadzadeh and Hartmann (2018a)* [68], we propose an Ant Colony based algorithm to solve a version of the problem including time constraints and detour possibility. It is statistically proved that this algorithm with the well-tuned parameters can provide good results for instances with up to 500 vehicles on the Swedish, grid-shaped and a series of random road network graphs. Furthermore, in our last published contribution in this field, *Nourmohammadzadeh and Hartmann (2018b)* [69], a Particle Swarm Optimisation (PSO) approach is proposed to tackle some FEP samples with multiple speeds on the Chicago network. This PSO, which differs from the one used in this thesis mostly in terms of handling the constraints, can

[1] General Algebraic Modelling System
[2] http://www.gurobi.com/

provide appropriate results for instances with up to 1000 vehicles. The solutions for smaller sizes are very near to the results obtained with Gurobi [36] solver in GAMS.

A considerable modelling attempt is done in *Zhang et al. (2017)* [100]. They formulate a platoon coordination problem with soft time windows as a mixed-integer linear programming problem and solve it with some exact methods. Their objective function consists of operation costs, schedule miss penalties and fuel costs. In the numerical example, a Swedish highway network model is used and the computation result shows that for 21 vehicles, the total cost can be reduced by 3.5% when the optimal preferred arrival times are chosen.

Recently, *Boysen et al. (2018)* [14] investigate a basic scheduling problem for the platoon building process on a single path. By changing some problem characteristics like the objective function, they derive different problem settings and provide an analysis of computational complexity for each. A couple of algorithms are proposed and applied to explore the impact of the diffusion of platooning technology, the maximum platoon length, and the tightness of time windows.

In another very recent work, *Luo et al. (2018)* [65] present a model for the FEP problem in which multiple speed options are considered for vehicles. They numerically test the performance of this model on a grid network and the Chicago-area highway network, which are very similar to those used in this thesis. A heuristic decomposed approach is proposed that applies a clustering algorithm to partition the set of vehicles, and then, routes each group separately.

Among this small number of works, the attempt into presenting mathematical formulations and specially applying meta-heuristic methods has been, however, little.

2.3 Summary

In this chapter, a review of the platooning field was presented. This shows that although numerous works connected to the platooning subject have been done mostly during the two past decades, there are only few number of researches into fuel-optimal platooning. Most of them have been presented in recent years. Therefore, further attempts in the fuel efficient platoon scheduling to derive new models containing more real elements as well as new applicable and fast solution approaches are strongly in demand.

Chapter 3

Models and Test Problems

In this chapter, the considered Fuel Efficient Platooning (FEP) problem is modelled from two different aspects resulting in two mixed-integer linear models. In the first model, decision variables of making platoons determine if two Heavy Duty Vehicles (HDVs, trucks or also simply called vehicles in this thesis) drive together at the same time and speed. However, the second model builds platoons by deciding if for any pair of vehicles, the latter one directly follows the other. Both of these conditions indicate that the two vehicles are together a platoon or a part of a larger platoon. These two models are both based on the same assumptions which match with the reality more than the previous models presented in the literature. We consider a set of allowable speeds for vehicles, moreover, their departure from the origin and arrival at the destination are timely constrained. In this chapter: firstly, the assumptions are presented in Section 3.1, and then, the two models are explained in details in Sections 3.2 and 3.3, respectively. In the following, the generation of the required test problems used in this thesis are explained in Section 3.4. It consists of two parts, namely: road networks and vehicles' data. Finally, the chapter is concluded in Section 3.5.

3.1 Assumptions

The FEP problem is here modelled based on more real-life conditions. However, some simplifications remain in order to keep the model tractable. Nonetheless, they are justifiable and in accordance to many real transportation systems. The assumptions or properties of our FEP problem, which are used in both of the following models are as below:

- The Heavy Duty Vehicles (HDVs or trucks) are all the same in terms of size and specifications.

- Although the air drag reduction exists for all vehicles of a platoon as Figure 3.1 shows its amounts for three consecutive HDVs, for simplification, the little air drag reduction or saving that the first HDV gains is ignored. In addition, due to minor differences in the air drags that the following HDVs experience, it is assumed that they all benefit from the same fuel saving factor of platooning. These assumptions exist in most of the previous works.

- Despite most of the previous works, we consider an earliest departure and latest arrival time (deadline) for any truck that help the model be more realistic because the goods should be ready for transportation and the delivery to destination must not be much delayed.

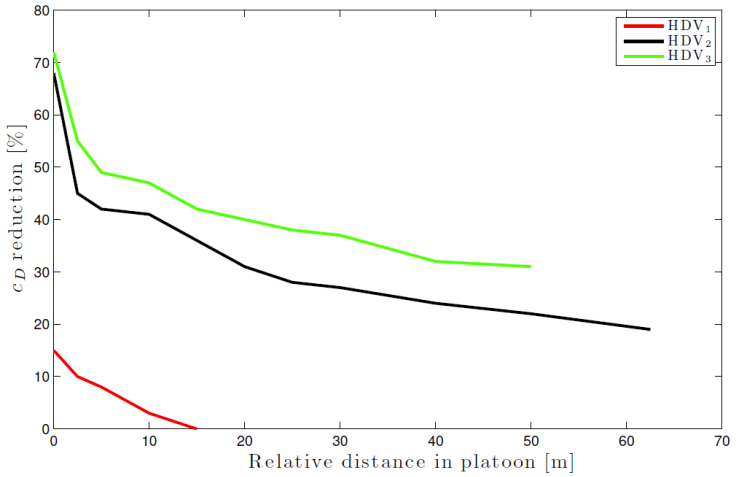


FIGURE 3.1: Air drag reduction for three consecutive Heavy Duty Vehicles (HDVs or trucks) driving as a platoon based on their relative distance. This is used from *Alam (2014)* [3] and is also available in *Kammer (2013)* [43]. However, it is originally presented by *Heinrich 1998* [95].

- Platoons are shaped in undirected graphs representing road networks. The graphs consist of some nodes as locations and some edges as road connections between them.

- The saving factor resulting from platooning depends on the speed. Faster a platoon drives, more fuel can be saved by platooning (*Luo et al.* [65]).

- The unit fuel cost and the time required to traverse one unit of distance depend on the speed. Faster an HDV drives, more fuel is used and shorter is the travelling time (*Luo et al.* [65]).

- The networks are scaled so they can be used easier and the calculations are simpler. However, this does not invalidate the models because the resulting costs can be multiplied by the real distance and fuel price of one unit. So the real cost can be obtained accordingly.

- A maximum allowable detour distance that a truck can drive over its shortest path(s) length is considered. It means that no truck can choose an alternative route, which is more than a specific amount longer than its shortest path(s).

3.2 Model 1

Modelling of the FEP problem is based on two groups of decision variables. The first group plays a more important role and builds the platoons and determines the routes of HDVs from their starting node to destination, while the second group are responsible for time scheduling. The first model is built based on main decision variables that determine if a pair of trucks drive together as a platoon or not. This model is based on the general mathematical structure introduced in *Larsson et al. (2015)* [54], whereas our version encompasses many new elements and constraints. The model is

presented in the two following subsections of explaining the notations and mathematical formulations.

3.2.1 Notations

The notation or elements used in the model are as below:

Inputs

H	The planning horizon
V	The set of trucks (HDVs or vehicles)
N	The set of nodes of the road network graph
E	The set of all available edges between the nodes
S	The set of allowable speeds
e_{ij}	The edge between node i and j
l_{ij}	The length of edge e_{ij}
O^v	The origin or starting node of truck v
D^v	The destination node of truck v
η_s	The fuel saving factor of platooning at speed s
β_s	The fuel cost per unit of distance if the truck drives at speed s
δ_s	The distance that a truck can traverse within a unit of time at speed s
T_e^v	The earliest time that truck v can leave its origin
T_{max}^v	The deadline of truck v to reach its destination no later than then
MD^v	Maximum allowable detour of truck v over its shortest path(s)
SP_{ij}	The shortest path length between nodes i and j
B_i^v	An auxiliary element showing if node i is the origin, destination or another point on the way of truck v
M	An auxiliary element containing a large value which can be considered equal to the maximum of deadlines to help in the model

Variables

x_{ijs}^v	Binary variable=1, if truck v traverses edge e_{ij} at speed s
t_{ij}^v	The time that truck v starts traversing edge e_{ij}
p_{ij}^{vw}	Binary variable=1, if truck v and w drive as a platoon on edge e_{ij}
α_{ij}^v	Binary variable=1, if truck v is the leader of a platoon on edge e_{ij}
c_{ij}	Unit fuel cost that is incurred on edge e_{ij}
z_{ij}^{vw}	An auxiliary element equal to an integer value which is equal or greater than the difference between the traversal time of trucks v and w on edge e_{ij}
y_{ij}^{vw}	An auxiliary binary variable to help in the model

3.2.2 Formulations

The mathematical formulations of Model1 are as below and they are explained underneath:

Objective

$$Min\ Z = \sum_{e_{ij} \in E} c_{ij} l_{ij} \tag{3.1}$$

Constraints

$$\sum_{s\in S} x_{ijs}^{v} \leq 1 \quad \forall v \in V; i,j \in N; e_{ij} \in E \tag{3.2}$$

$$B_i^v = \begin{cases} -1 & \text{if } i = O^v \\ 1 & \text{if } i = D^v \\ 0 & otherwise \end{cases} \quad \forall v \in V; i \in N \tag{3.3}$$

$$\sum_{h:e_{hi}\in E} \sum_{s} x_{his}^{v} - \sum_{j:e_{ij}\in E} \sum_{s} x_{ijs}^{v} = B_i^v \quad \forall v \in V; i \in N \tag{3.4}$$

$$M = \max_{v\in V}\{T_{max}^{v}\} \tag{3.5}$$

$$t_{O^v j}^{v} \geq T_e^v - M(1 - \sum_{s} x_{O^v js}^{v}) \quad \forall v \in V; j \in N; e_{O^v j} \in E \tag{3.6}$$

$$t_{ij}^{v} - t_{hi}^{v} - M(\sum_{s\in S} x_{ijs}^{v} + \sum_{r\in S} x_{hir}^{v}) \geq \sum_{r\in S} \frac{l_{hi}}{\delta_r} x_{hir}^{v} - 2M \quad \forall v \in V; \\ h,i,j \in N; e_{ij}, e_{hi} \in E \tag{3.7}$$

$$-z_{ij}^{vw} \leq t_{ij}^{v} - t_{ij}^{w} \leq z_{ij}^{vw} \quad \forall v,w \in V; v < w; i,j \in N; e_{ij} \in E \tag{3.8}$$

$$M(1 - p_{ij}^{vw}) + z_{ij}^{vw} \geq 0 \quad \forall v,w \in V; v < w; i,j \in N; e_{ij} \in E \tag{3.9}$$

$$M(1 - p_{ij}^{vw}) - z_{ij}^{vw} \geq 0 \quad \forall v,w \in V; v < w; i,j \in N; e_{ij} \in E \tag{3.10}$$

$$2p_{ij}^{vw} - (x_{ijs}^{v} + x_{ijs}^{w}) \leq 0 \quad \forall v,w \in V; v < w; i,j \in N; e_{ij} \in E; \\ s \in S \tag{3.11}$$

$$p_{ij}^{vw} \geq (x_{ijs}^{v} + x_{ijs}^{w}) + z_{ij}^{vw} - My_{ij}^{vw} - 1 \quad \forall v,w \in V; v < w; i,j \in N; \\ e_{ij} \in E; ; s \in S \tag{3.12}$$

$$p_{ij}^{vw} \geq (x_{ijs}^{v} + x_{ijs}^{w}) - z_{ij}^{vw} - M(1 - y_{ij}^{vw}) - 1 \quad \forall v,w \in V; v < w; \\ i,j \in N; e_{ij} \in E; s \in S \tag{3.13}$$

$$\alpha_{ij}^{w} + \sum_{v=1}^{w-1} p_{ij}^{vw} \geq \sum_{s\in S} x_{ijs}^{w} \quad \forall w \in V; i,j \in G; e_{ij} \in E \tag{3.14}$$

$$\alpha_{ij}^{v} \leq \sum_{s \in S} x_{ijs}^{v} \quad \forall v \in V; i, j \in N; e_{ij} \in E \tag{3.15}$$

$$\alpha_{ij}^{w} \leq 1 - p_{ij}^{vw} \quad \forall v, w \in V; v < w; i, j \in N; e_{ij} \in E \tag{3.16}$$

$$t_{iD^v}^{v} + \sum_{s \in S} \frac{l_{iD^v}}{\delta_s} . x_{iD^v s}^{v} \leq T_{max}^{v} \quad \forall v \in V; i \in N; e_{iD^v} \in E \tag{3.17}$$

$$\sum_{e_{ij} \in E} \sum_{s \in S} l_{ij} x_{ijs}^{v} \leq SP_{O^v D^v} + MD^{v} \quad \forall v \in V \tag{3.18}$$

$$M \sum_{s \in S} x_{ijs}^{v} \geq t_{ij}^{v} \quad \forall v \in V; i, j \in N; e_{ij} \in E \tag{3.19}$$

$$c_{ij} = \sum_{v \in V} \sum_{s \in S} \beta_s [\alpha_{ij}^{v} + (1 - \eta_s)(x_{ijs}^{v} - \alpha_{ij}^{v})] \quad \forall i, j \in N; e_{ij} \in E \tag{3.20}$$

$$x_{ijs}^{v}, p_{ij}^{vw}, \alpha_{ij}^{vw} \in \{0, 1\}; z_{ij}^{vw} \in \mathbb{Z}, , t_{ij}^{v} \in \mathbb{R}_{\geq 0}$$

Equation (3.1) is the objective function of our problem that accumulates the whole unit fuel cost incurred on all edges.

Constraint (3.2) enforces that each vehicle can drive at most at one speed from the set of allowable speeds throughout each edge. Relation (3.3) assigns the correct value to B_i^v based on if it is the origin (=-1), destination (=1) or another node along the path of truck v or generally of the graph (=0). This auxiliary element is used in the next constraint. Constraint (3.4) is responsible for node balancing in the system. It ensures that each truck leaves its origin, reaches its destination, and if a truck enters a node which is neither the origin nor the destination, it then quits this node to travel towards its destination. This prevents any vehicle from being vanished or emerging in the middle of the way in the system. Equation (3.5) sets M equal to the largest deadline among all vehicles to help in the next constraints of the model. Constraint (3.6) respects the earliest departure time from the origin for any vehicle. It becomes active only if truck v chooses edge $e_{O^v j}$, i.e $\sum_s x_{O^v js}^{v} = 1$.

Constraint (3.7) schedules the traversal time of the vehicles along any two consecutive edges. It makes the succeeding traversal time be equal or later than proceeding traversal time plus the required time to pass the edge based on the chosen speed r. If there is no waiting time at node i, then the two sides of this constraint are equal.

To reduce the very large number of variables resulting from the symmetry, the variables containing both of the indices v and w, i.e. z_{ij}^{vw}, p_{ij}^{vw} and y_{ij}^{vw}, are only defined for the case that $v < w$.

Constraints (3.8-3.13) provide the relation between p_{ij}^{vw}, x_{ijs}^{v}, x_{ijs}^{w}, t_{ij}^{v} and t_{ij}^{w}. Constraint (3.8) sets z_{ij}^{vw} equal to an integer value greater than or equal to the difference between the traversal time of v and w along edge ij. However, its main function is to force $t_{ij}^{v} = t_{ij}^{w}$ if $z_{ij}^{vw} = 0$. (3.9) and (3.10) ensure that if p_{ij}^{vw}=1, then t_{ij}^{v} and t_{ij}^{w} must be equal based on z_{ij}^{vw}=0 and (3.8). (3.11) makes both x_{ijs}^{v} and x_{ijs}^{w} equal to 1 if p_{ij}^{vw}=1. The two constraints (3.12) and (3.13)

ensure that if t_{ij}^v=t_{ij}^w and x_{ijs}^v=$x_{ijr}^w = 1$, then p_{ij}^{vw}=1. y_{ij}^{vw} is a binary variable that makes one of (3.9) and (3.10) become trivially satisfied for v, w and e_{ij}. An integer value for z_{ij}^{vw} is required here because if we use $t_{ij}^v - t_{ij}^w$ instead of z_{ij}^{vw} like in *Larsson et al. (2015)* [54], in one worst case that $x_{ij}^v = x_{ij}^w = 1$ and $t_{ij}^v - t_{ij}^w \in (-1, 1)$, the problem will be infeasible because p_{ij}^{vw} must be 1, which contradicts (3.9) and (3.10).

It is here assumed that the truck with the least index is the leader. Hence, constraints (3.14)-(3.16) set $\alpha_{ij}^w = 1$ if and only if no truck with a smaller index platoons with i. Constraint (3.14) makes α_{ij}^w equal to 1 if x_{ij}^w=1 and for all trucks with a smaller index, i.e. $v < w$, p_{ij}^{vw}=0. If vehicle v does not traverse e_{ij}, α_{ij}^v must be zero, that is fulfilled by constraint (3.15). Constraint (3.16) forces all p_{ij}^{vw} to be zero if $\alpha_{ij}^w = 1$.

Constraint (3.17) ensures that every truck reaches its destination before the deadline. Likewise, respecting the maximum allowable detour (MD^v) is guaranteed by constraint (3.18) .

If truck v do not traverses edge e_{ij}, the corresponding traversal time must be zero, which is enforced by constraint (3.19).

Equation (3.20) is an important part in the model, which calculates all the unit fuel cost incurred on each edge e_{ij}. This calculation is done based on the trucks' speed and the corresponding fuel consumption per unit of distance β_s, whether the truck is a leader (or travelling alone) or a follower determined by α_{ij}^v, and the corresponding saving factor of platooning η_s.

This model is called Model1 in this thesis. The model presented in *Larsson et al. (2015)* [54] is a special case of this model where the deadlines are all shifted to the end of planning horizon, the earliest departure times are all zero, the set of allowable speeds contains only one element or cruise speed, and the maximum detours are large enough . So the transformation of Model1 to that model is possible by applying the below algebraic rules and corresponding modifications:

-$T_{max}^v = H,\ \forall v \in V$
-$T_e^v = 0,\ \forall v \in V$
-$S = s_3$, see 3.4.2
-$MD^v = \max_{q \in Fl^v}\{q\} - SP_{O^v D^v},\ \forall v \in V$, where Fl^v is a set including the lengths of feasible routes of v

3.3 Model 2

In this section, we present a different model which constructs platoons by checking if for any two vehicles one directly follows the other on a specific edge at the same time and speed. If all of these conditions hold, they drive in the same platoon on that edge. This approach leads to a simpler model in comparison to Model1, however, both models have many variables and some constraints in common. Hence, in Subsection 3.2.1, only the new elements used in Model2 are explained. This model is build based on the models given in *Larson et al. (2016)* [52] and *Luo et al. (2018)* [65].

3.3.1 Notations

The specific notations of Model2 in comparison to Model1 are only two variables listed below and the rest of elements presented in Subsection 3.2.1 are here applicable as well.

Variables

f_{ijs}^{wv}	Binary variable=1, if truck (vehicle) w directly follows truck v on edge e_{ij} at speed s
WT_i^v	The waiting time of truck v in node i

3.3.2 Formulations

Although there are just few specific variables in the second model, the modelling approach is different and based on directly following a front vehicle with the same speed. Nonetheless, some fundamental relations such as node balancing exist in both models.

$$Min\ Z = \sum_{e_{ij} \in E} c_{ij} l_{ij} \tag{3.21}$$

$$\sum_{s \in S} x_{ijs}^v \leq 1 \quad \forall v \in V; i, j \in N; e_{ij} \in E \tag{3.22}$$

$$B_i^v = \begin{cases} -1 & \text{if } i = O^v \\ 1 & \text{if } i = D^v \\ 0 & otherwise \end{cases} \quad \forall v \in V, i \in N \tag{3.23}$$

$$\sum_{h:e_{hi} \in E} \sum_s x_{his}^v - \sum_{j:e_{ij} \in E} \sum_s x_{ijs}^v = B_i^v \quad \forall v \in V; i \in N \tag{3.24}$$

$$M = \max_{v \in V} \{T_{max}^v\} \tag{3.25}$$

$$-M(1 - \sum_{s \in S} f_{ijs}^{wv}) \leq t_{ij}^w - t_{ij}^v \leq M(1 - \sum_{s \in S} f_{ijs}^{wv}) \quad \forall v, w \in V; v < w, \\ i, j \in N; e_{ij} \in E \tag{3.26}$$

$$\sum_{v:v<w} f_{ijs}^{wv} \leq 1 \quad \forall w \in V; i, j \in N; e_{ij} \in E; s \in S \tag{3.27}$$

$$\sum_{w:w>v} f_{ijs}^{wv} \leq 1 \quad \forall v \in V; i, j \in N; e_{ij} \in E; s \in S \tag{3.28}$$

$$2f_{ijs}^{wv} \leq x_{ijs}^v + x_{ijs}^w \quad \forall v, w \in V; v < w; i, j \in N; e_{ij} \in E; s \in S \tag{3.29}$$

$$-M(1 - \sum_{s \in S} x_{O^v js}^v) \leq t_{O^v j}^v - T_e^v - WT_{O^v}^v \leq M(1 - \sum_{s \in S} x_{O^v js}^v) \\ \forall v \in V, j \in N, e_{O^v j} \in E \tag{3.30}$$

$$-M(1-\sum_{s\in S} x^v_{iD^v s}) \leq T^v_{max} - t^v_{iD^v} - WT^v_{D^v} - \sum_{s\in S} \frac{l_{iD^v}}{\delta_s} x^v_{iD^v s} \leq M(1-\sum_{s\in S} x^v_{iD^v s})$$
$$\forall v \in V; i \in N; e_{iD^v} \in E \quad (3.31)$$

$$-M(2-\sum_{r\in S} x^v_{hir} - \sum_{s\in S} x^v_{ijs}) \leq t^v_{hi} - t^v_{ij} - WT^v_i - \sum_{r\in S} \frac{l_{hi}}{\delta_r} x^v_{ijs} \leq$$
$$M(2-\sum_{r\in S} x^v_{hir} - \sum_{s\in S} x^v_{ijs})$$
$$v \in V; h,i,j \in N; e_{hi}, e_{ij} \in E; i \neq O^v, D^v \quad (3.32)$$

$$M\sum_{s\in S} x^v_{ijs} \geq t^v_{ij} \quad \forall v \in V; i,j \in N; e_{ij} \in E \quad (3.33)$$

$$WT^v_i \leq M(\sum_{j:e_{ij}\in E}\sum_{s\in S}\sum_{r\in S} x^v_{ijs} + x^v_{jir}) \quad \forall v \in V; i \in N \quad (3.34)$$

$$\sum_{e_{ij}\in E}\sum_{s\in S} l_{ij} x^v_{ijs} \leq SP_{O^v D^v} + MD^v \quad \forall v \in V \quad (3.35)$$

$$c_{ij} = \sum_{w\in V}\sum_{s\in S} \beta_s (x^w_{ijs} - \eta_s \sum_{v:v<w} f^{wv}_{ijs}) \quad \forall i,j \in N; e_{ij} \in E \quad (3.36)$$

$$x^v_{ijs}, p^{vw}_{ij}, \alpha^{vw}_{ij}, f^{wv}_{ijs} \in \{0,1\}; z^{vw}_{ij} \in \mathbb{Z}, t^v_{ij}, WT^v_i \in \mathbb{R}_{\geq 0}$$

Equation (3.21) is the objective function which sums up the total cost incurred on all edges. Constraint (3.22) is exactly (3.2) of the Model1. Constraints (3.23)-(3.24) are node balancing relations for each truck from the origin to destination, which are the same as (3.3)-(3.4).

A large value for M is assigned by (3.25) like in the previous model. Constraint (3.26) ensures that if truck w directly follows truck v on edge ij, i.e. $\sum_{s\in S} f^{wv}_{ijs} = 1$, then they must have the same traversal time, i.e. $t^v_{ij} = t^w_{ij}$. Like in Model 1, here we reduce a considerable amount of complexity by removing the symmetry through assuming $v < w$ in f^{wv}_{ijs}.

Each vehicle can directly follow no more than one other vehicle. This is ensured by constraint (3.27). Similarly, constraint (3.28) guarantees that each vehicle can be followed by at most one other vehicle. Constraint (3.29) states if vehicle w directly follows vehicle v on edge e_{ij} with speed s, i.e. $\sum_{s\in S} f^{wv}_{ijs}$=1, then both of the vehicles must traverse e_{ij} at speed s, that means $x^v_{ijs} = x^w_{ijs} = 1$.

Constraint (3.30) ensures that if truck v leaves its origin for node j, i.e. $\sum_{s\in S} x^v_{O^v js}$=1, then the traversal time of edge $e_{O^v j}$, i.e. $t^v_{O^v j}$, is equal to earliest departure time (T^v_e) plus the waiting time at the beginning in the origin, $WT^v_{O^v}$. Constraint (3.31) states that in case vehicle v reaches its destination through edge e_{iD^v} ($\sum_{s\in S} x^v_{iD^v s} = 1$), the traversal time plus the required time to traverse the edge and waiting time or earliness at the destination is equal to the deadline. Constraint (3.32) states that the

difference between two consecutive traversal times is equal to required time to pass the first edge ($\sum_{r \in S} \frac{l_{hi}}{\delta_r} x^v_{ijs}$) plus the waiting time in the node in-between (WT^v_i).

If these is no traversal along edge e_{ij}, the corresponding traversal time must be zero, this is assured by (3.33). Constraint (3.34) insures that if vehicle v does not pass through node i, then the corresponding waiting time is zero. Respecting the maximum detour is done by (3.35).

Equation (3.36) sums up the unit costs incurred by vehicles that traverse edge e_{ij}. Based on whether the vehicle is directly following another vehicle or not, i.e. if the platooning discount should be considered or not, this cost is computed for each vehicle.

The transformation of Model2 to the one of *Luo et al. 2018* [65] is possible by considering the maximum detours equal to the longest feasible route length of the vehicle minus the shortest path length.
-$MD^v = \max_{q \in Fl^v}\{q\} - SP_{O^v D^v}$, $\forall v \in V$, where Fl^v is a set including the lengths of feasible routes of v

3.4 Test Problems

In the next step, we need to generate some test instances to examine the goodness of our solution methodologies, which are proposed in the next chapters. A test problem of FEP includes two parts, firstly, the inputs related to the road network graph, and secondly, the ones of vehicles. In this section, it is explained how we provide both input groups. Six different road graphs are used, which are explained in 3.4.1. In the following, the generation of specific data related to vehicles, which travel on these road networks are described in 3.4.2.

3.4.1 Road network graphs

The network graphs are the vital part of input data because the vehicle platooning occurs on them. Various graphs should be used in order to examine the capability of our solution methodologies in different cases. For this sake, we use both real based and some worse case hard graphs.

6 different graphs are altogether used. Three of them are real based and mostly used in the literature: Chicago highways network (*Larson et al, 2016* [52]; *Luo et al., 2018* [65]), German (*Larsson et al., 2015* [54]; *Kammer, 2013* [43]) and Swedish road network (*Kammer, 2013* [43], *Zhang, 2017* [100]). However, these networks are quite different in this thesis compared to those used in the literature because of the included nodes and edges, and also the applied scaling. Three grid network graphs of different sizes, where there are many alternative routes for each vehicle, are chosen as well. Platooning on such networks is done in *Larson et al. (2016)* [52] and *Luo et al. (2018)* [65].

Highway network of the Chicago area

The network of highways in Chicago area in the USA is the simplest real based graph that is used in this thesis. As platooning is not beneficial at slow speeds, only highways are considered and the main locations are included as nodes in this network. The distances are quite short between the nodes so that in our scaling one unit of distance is approximately equal to 1km.

This graph, which includes altogether 163 nodes and 165 edges, is shown in figure 3.2.

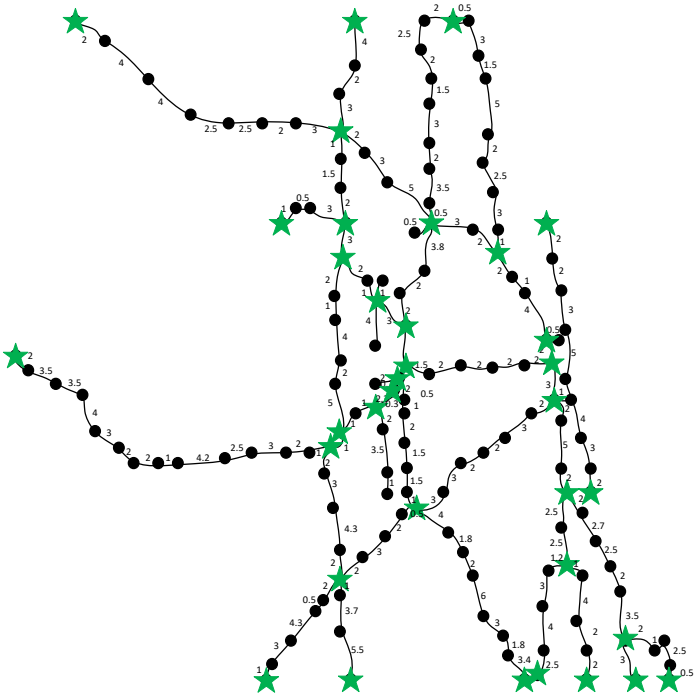


FIGURE 3.2: The Chicago road network. The star nodes have more priority to be the destinations

German autobahn network

The German Autobahn network used here comprises important cities or locations in Germany as nodes and the highway connections between them. There are 225 nodes and 292 edges in this network. Each unit is assumed to be about 15km. This graph is depicted as Figure 3.3

Swedish road network

The third network is the most complex real based one to conduct our experiments on. This is derived from the real road network and main locations in Sweden. This graph of Swedish road network is shown in Figure 3.4. It contains 356 nodes and 408 edges, and the scaling is according to one unit equal to approximately 12km.

Grid networks

Besides the three real based networks, the performance of our algorithms should be tested with some artificial networks which make the platooning computation more complex. Grid networks are very complicated graphs for vehicle platooning due to so many equal routing options that each vehicle has from its origin to destination. A 10×10 Grid network is shown in Figure 3.5. Each edge is considered to be equal to one unit of distance. If for a vehicle the coordinates of the origin and destination are (x_1, y_1) and

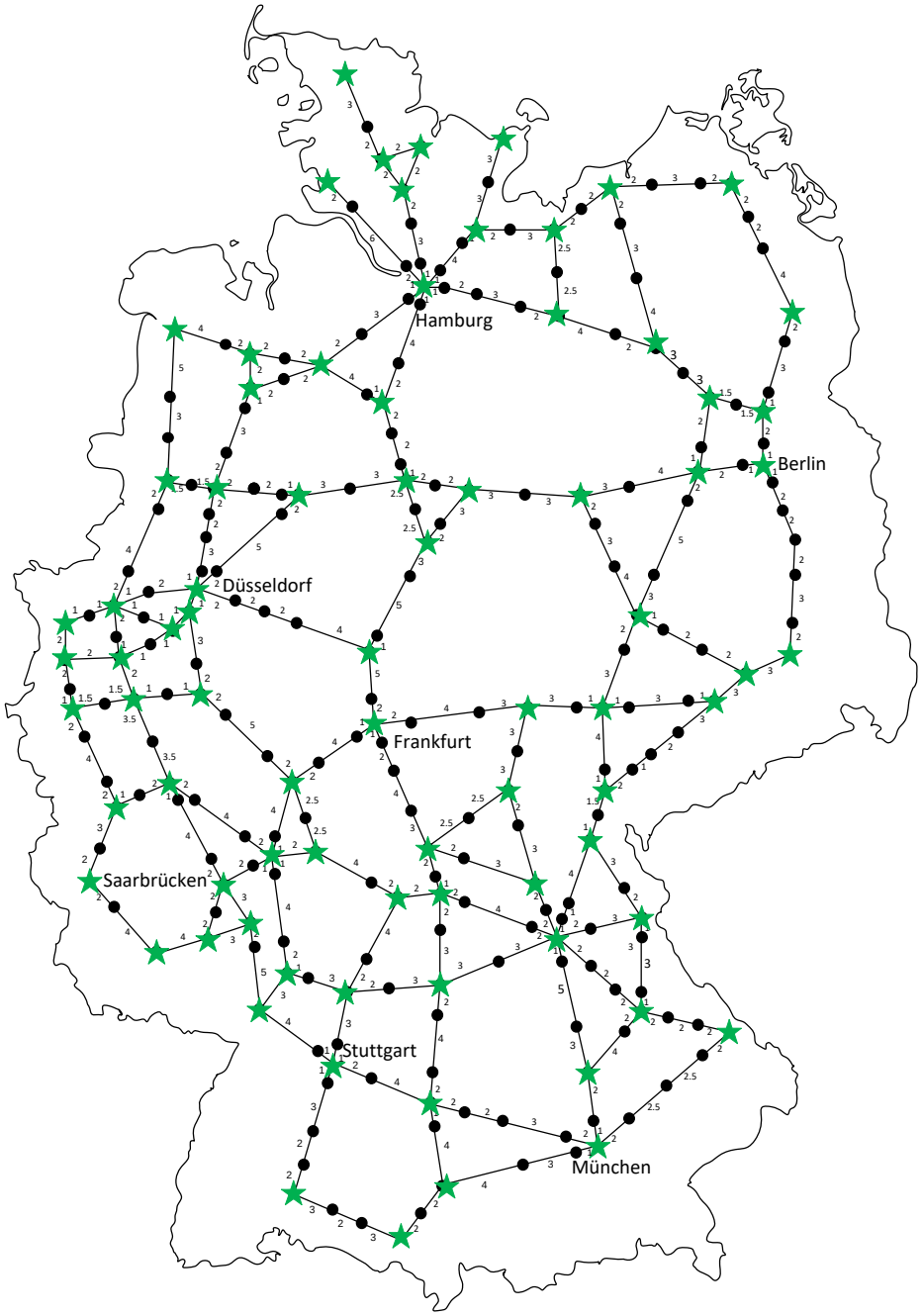


FIGURE 3.3: The German autobahn network. The star nodes have more priority to be the destinations

(x_2, y_2), the number of the available equal shortest paths from the origin to destination are $\binom{(x_2-x_1)+(y_2-y_1)}{x_2-x_1}$. In addition, if the deadline and maximum detour allow, there are some other equal routes with a longer length. Thus, each vehicle has many alternative paths and these result in more platooning opportunity for the whole system. This increases the computational effort for any FEP problem on such Grid networks.

In this thesis, we use Grid networks of sizes 10×10 including 100 nodes and 180 edges, 30×30 including 900 nodes and 1740 edges, and 50×50 including 2500 nodes and 4900 edges.

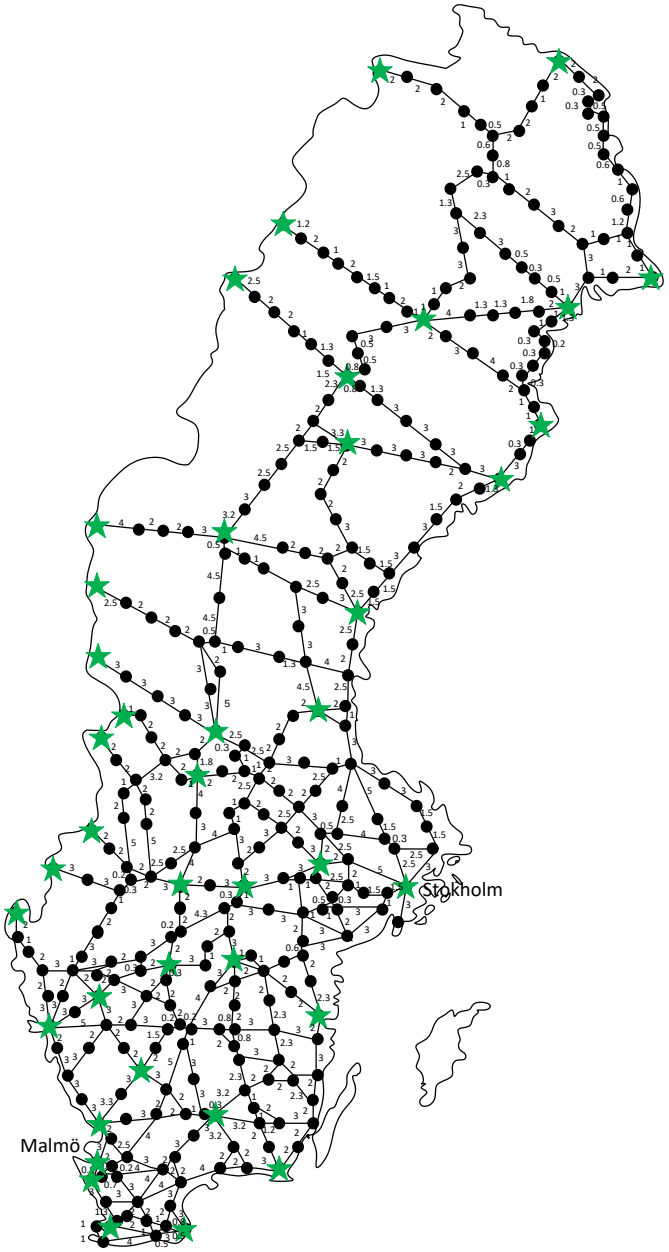


FIGURE 3.4: The Swedish road network. The star nodes have more priority to be the destinations

3.4.2 Vehicle Data

The data of vehicles, including: origin (O^v), destination (D^v), earliest release time (t_e^v), deadline (T_{max}^v), and maximum allowable detour (MD^v) of each vehicle as well as their specifications such as fuel saving factor of platooning η_s, fuel cost per unit of distance β_s and the traversable distance in a unit of time δ_s are absolutely important inputs.

As we do not have access to real data of any transportation company, these truck data are generated randomly but to some extend according to real patterns. There are some significant points to respect for creating the vehicle information. Firstly, the data have to conform with reality as mentioned. Secondly, they must not hinder the feasibility of the problem,

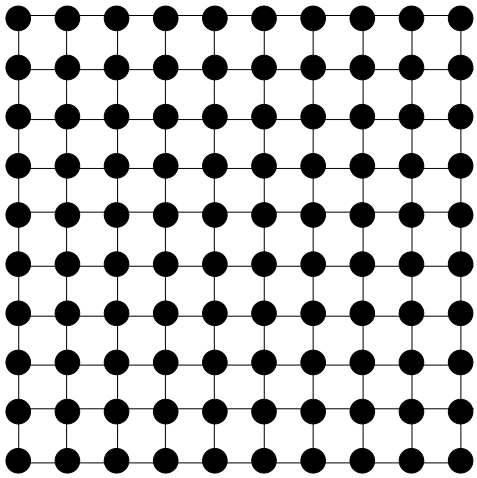


FIGURE 3.5: The Grid10 network.

TABLE 3.1: Parameters related to the three allowable speeds

Speed Options	s_1	s_2	s_3
miles per hour	75	65	55
η_s	0.15	0.13	0.10
β_s	1.00	0.93	0.77
δ_s	1.36	1.15	1.00

and finally, they should not be so loose that ease the problem too much or make some constraints be satisfied in any case.

As for vehicle speeds, it is assumed that all trucks can drive at only three allowable speeds. Then, we use Table 3.1, which is according to the information provided in *Luo et al. (2018)* [65].

The following method is applied to generate the mentioned vehicles' inputs: the origins of trucks are randomly generated from all of the graph nodes. Contrarily, the destination of trucks are spread within a limited number of nodes equal to $\lfloor 0.3 \times |V| \rfloor$ to provide more platooning opportunity. Firstly, the hub or main nodes (shown with stars) located at the borders are chosen as destinations. It is due to the fact that there are large volumes of transportation into outside through such end nodes. If we still need to have more destinations, the middle star nodes, and finally, normal nodes are selected. For the Gird networks, firstly, the nodes which are on the perimeter of the whole square, and then, any other node can be chosen as a destination. The planning horizon H or maximum of time is assumed to be a sufficiently large value equal to the smallest integer greater or equal than the sum of the lengths of all edges. The earliest departure (release) times of vehicles are randomly generated in the interval $[0, H - SP_{O^v D^v}]$. Subsequently, deadlines are uniformly generated in $[T_e^v + SP_{O^v,D^v}, H]$. Since the saving factor of platooning at the cruise speed is 0.1, a vehicle does not choose any route which is more than 0.1 of the shortest path(s) length longer than the shortest path(s). This is because of more expenses that will be imposed to the system

in comparison with choosing the shortest path(s) even if the vehicle can drive in platoons on the whole of the longer route. Therefore, any amount which is greater than $SP_{O_v,D_v} \times 0.1$ as MD^v makes the related constraint, i.e 3.18 in Model1 and 3.35 in Model2, be excessive. However, choosing a small amount for MD^v decreases platooning opportunities. Thus, here we assume that $MD^v = 0.09 \times SP_{O_v,D_v}$.

The rules of generating vehicles' data are also summarised in Table 3.2.

TABLE 3.2: Generation of Trucks' data

Truck data	Generation rule
H	$\lceil \sum_{e_{ij} \in E} l_{ij} \rceil$
O^v	Randomly chosen from the graph nodes
D^v	Randomly chosen from $\lfloor 0.3 \times \lvert V \rvert \rfloor$ nodes; firstly border star nodes (for Grid networks those on the perimeter), then other star nodes, and finally, normal nodes
T_e^v	$U[0, H - SP_{O^v D^v}]$
T_{max}^v	$U[T_e^v + SP_{O^v,D^v}, H]$
MD^v	$0.09 \times SP_{O^v D^v}$

3.5 Summary

In this chapter, two main models for the considered FEP problem were proposed. Two different modelling concepts are used to make them. Some other approaches of modelling have been also presented in *Kammer (2013)* [43] and *Nourmohammadzadeh and Hartmann (2016)* [70] based on discrete time. However, these approaches entail too many variables in the model and this considerably increases the complexity of solving an FEP problem. In the next chapter, the presented models are exactly solved and their results are compared.

Chapter 4

Exact Solutions

Following the introduced mathematical models and problem instance generation method, in this chapter, the results of coding both models in GAMS and solving instances of different scales by the exact solver of CPLEX are presented. Since the solver ability reduces rapidly by slight increases in the number of vehicles, some alternative strategies are proposed to lessen the problem complexity and their influence are examined. The chapter is organised as follows: the exact results based on the two models of the previous chapter are given in Section 4.1. Section 4.2 proposes some methods to reduce the high complexity of an FEP problem. The results of the decomplexified models are presented in Section 4.3. The performance of the two models and their decomplexified versions are statistically compared to each other in Section 4.4. Finally, Section 4.5 gives a summary of the works done in this chapter.

4.1 Results of Model1 vs. Model2

The exact solution in this thesis has a procedure like this: firstly, the mathematical formulations of Model1 and Model2, which are defined in the previous chapter, should be coded in a platform (language). Then, the input data of each instance should be attached. Finally, an appropriate solver should be called to tackle each instance.

The General Algebraic Modelling System (GAMS) is a high-level modelling system (language) for mathematical programming and optimisation. It consists of a language compiler and a stable of integrated high-performance solvers. GAMS is tailored for complex large scale modelling applications, and allows you to build large maintainable models that can be adapted quickly to new situations. GAMS is specifically designed for modelling linear, nonlinear and mixed-integer optimisation problems [18].

One of the powerful optimisers available in GAMS is CPLEX. It is designed to solve large and difficult problems quickly and with minimal user intervention. CPLEX can be applied to linear, quadratically constrained and mixed-integer programming problems [31]. Since both of the models are linear, CPLEX is one of the best options to solve them with.

We start from the problem with 10 trucks and increase the vehicle set size to see up to which size it is possible to exactly solve the FEP problem. 20 instances for each size are generated on the 6 graphs of the previous chapter.

The fuel saving percentage resulting from solving each instance is calculated as:

$$\frac{Fuel\ cost\ of\ the\ platooning\ solution\ (objective\ function\ value)}{Shortest\ path\ fuel\ cost\ without\ any\ platooning} \times 100$$

The numerator of this fraction is the fuel consumption by the optimal platooning

solution or the objective function value of the models, whereas the denominator is the fuel consumption when all trucks travel on their shortest path(s) and there is no platooning effect.

A limiting factor for solving the FEP problem is the required solution time. Thus, it is very important for each method to obtain its final solution in a shorter time. In this thesis, each solver or solution methodology is allowed to deal with the problem for 30 minutes or 1800 seconds. After this time limit, the solution process is terminated and the best solution found is reported.

The mathematical formulations of Model1 and Model2 are coded in GAMS and solved with the CPLEX solver for each scale and on each of the 6 graphs, i.e. the Chicago, German, Swedish, Grid10, Grid30 and Grid50 network. All experiments of this thesis are executed in parallel on computers with an Intel(R) Core(TM) i7, 3.10GHz CPU and 16GB of RAM.

The saving percentage and solution time of 20 instances for each size and model are shown by box plots, so the capabilities of the two models can be visualised. Two Figures are dedicated to each network, one depicting the savings and one the execution times of solving.

These box plots are shown in Figures 4.1 to 4.12. The results for three sizes with 10, 20 and 50 trucks are depicted as S10, S20 and S50. From S50, the solver is completely unable to deal with any instance coded through any model within our time limit. Since the larger instances cannot be properly tackled by the exact solver, only these three sizes are shown.

In terms of saving, we also obtain upper bounds (shown as UB) of saving from the CPLEX solver. Therefore, as the first box plot for each size, the upper bounds obtained in the solution process of Model2 are shown.

When the time limit is reached, we show 1800s as the execution time although the problem has not been optimally solved. So wherever the box plots show 1800s, it means that the optimal solutions of the instances are not obtained within our time limit, and hence, the last solutions found are shown in the saving box plot.

In the following, some information about the number of obtained optimal solutions, average saving and execution time corresponding to each scale for any of the six networks is provided:

Figure 4.1 shows the saving percentages for the 20 instances of each size obtained from solving Model1 and Model2 on the Chicago network as well as the saving upper bounds. Figure 4.2 depicts the execution times. The number of obtained optimal results and the averages of savings and times are given in Table 4.1. For S10, Model2 can reach the optimal solutions of all the 20 instances, whereas Model1 can find 15 of them. The average saving and execution time of Model1 are 1.21% and 1680s, respectively. However, an average saving of 1.24% is obtained in a mean solution time of 690s through Model2, which is equal to its maximum possible saving. As we double the number of trucks and make S20 examples, the maximum possible saving grows to an average of 1.41% indicated by the upper bounds. Model1 cannot end with any optimal solution within 1800s but Model2 solves 13 instances to optimality in 1683s on average. When the number of trucks increases to 50, none of the two models can find any optimal solution. Thus, the last found solutions by reaching the time limit are considered instead. The average saving of Model1 and Model2 are equal to 1.93% and 2.09%, however, the upper bounds show a maximum average saving of 2.26%.

Figures 4.3 and 4.4 illustrate the results on the German network. Table 4.2 presents the results' summary. Since this network is wider and more complicated than the first one, the savings are lower and the execution times are longer for the same sizes. For S10 examples, Model1 obtains 13 optimal solutions and the average of all solutions is 1.11% and mean time is 1704s. Model2 is more capable because it can reach all of the optimal solutions with an average saving equal to 1.14% in a mean time of 693s. For S20 , Model1 ends up with no optimal solution and an average saving of 1.19%, while Model2 solves half of the instances to optimality in a mean time of 1726s. Its average saving is 1.35%. Nevertheless, the average

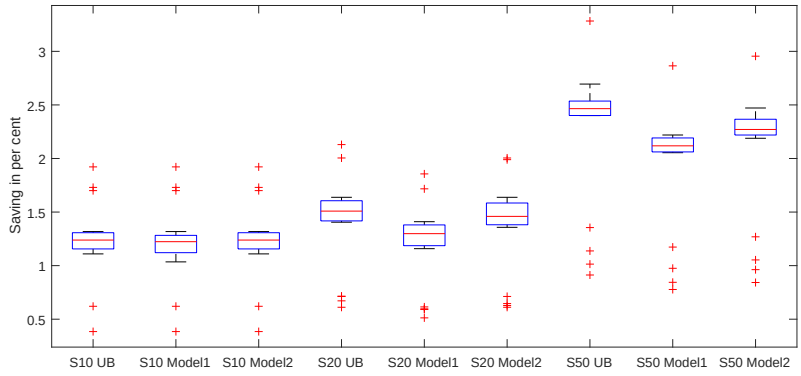


FIGURE 4.1: Saving for instances with 10, 20 and 50 trucks on the Chicago network

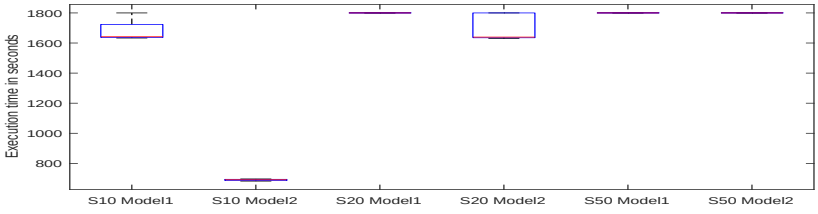


FIGURE 4.2: Execution time of solving for instances with 10, 20 and 50 trucks on the Chicago network

TABLE 4.1: The number of obtained optimal solutions, average savings and solution times for the Chicago network with Model1 and Model2

Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	15	20	1.21	1.24	1.24	1680	690
S20	0	13	1.21	1.37	1.41	1800	1683
S50	0	0	1.93	2.09	2.26	1800	1800

S20 UB is 1.40%. Turning to S50, as expected, the two Models are unable to find any optimal result. The average of upper bounds, Model1 and Model2 savings are 2.13%, 1.83% and 1.95%, respectively.

TABLE 4.2: The number of obtained optimal solutions, average savings and solution times for the German network with Model1 and Model2

Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	13	20	1.11	1.14	1.14	1704	693
S20	0	10	1.19	1.35	1.40	1800	1726
S50	0	0	1.83	1.95	2.13	1800	1800

Figures 4.5 and 4.6 show the results for the Swedish network, which is harder than the previous ones. Table 4.3 presents the means of saving and solution time. The results are as follows: for S10, Model1 achieves 12 optimal solutions, and the average saving and execution time are 0.91% and 1720s, respectively. On the other

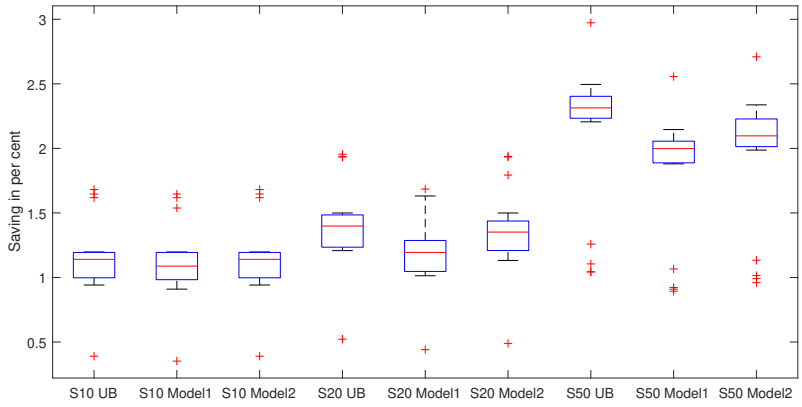


FIGURE 4.3: Saving for instances with 10, 20 and 50 truck on the German network

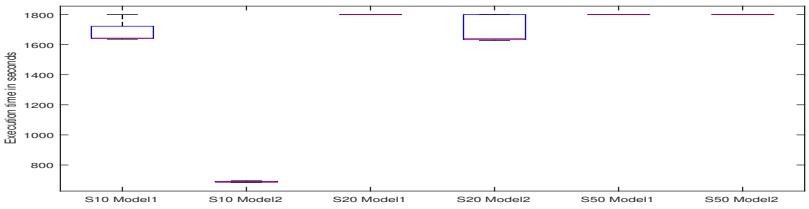


FIGURE 4.4: Execution time of solving for instances with 10, 20 and 50 trucks on the German network

hand, solving Model2 results in calculating all the optimal savings. Its average saving and execution time are 0.95% and 716s, respectively. For S20, the number of optimal solved instances by Model1 is 0 against 9 of Model2. The average saving of Model1 is 0.96%, whereas for Model2, it is 1.08 % in a mean time of 1739s. No optimal results are observed for S50 instances but the average of the best results through Model1 and Model2 are 1.67% and 1.82%, which considerably differ from the mean UB of 1.97%.

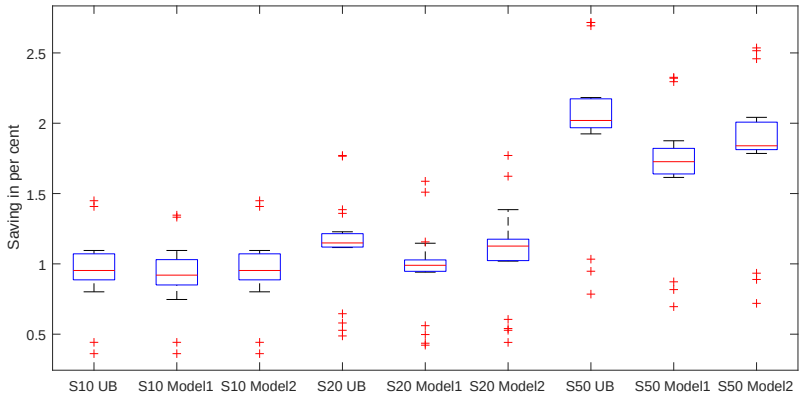


FIGURE 4.5: Saving for instances with 10, 20 and 50 trucks on the Swedish network

After analysing the results of real based networks, now we turn to grid networks, which provide more platooning opportunities. The smallest one is Grid10 with

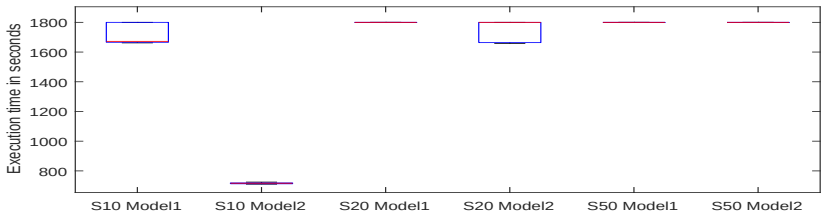


FIGURE 4.6: Execution time of solving for instances with 10, 20 and 50 trucks on the Swedish network

TABLE 4.3: The number of obtained optimal solutions, average savings and solution times for the Swedish network with Model1 and Model2

Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	12	20	0.91	0.95	0.95	1720	716
S20	0	9	0.96	1.08	1.12	1800	1739
S50	0	0	1.67	1.82	1.97	1800	1800

the savings and times shown by Figures 4.7 and 4.8, and means given in Table 4.4. Starting from S10, Model1 returns the optimal results of 10 instances. Its overall average saving and time are 1.15% and 1733s, respectively. However, Model2 gives the optimal results of all instances, which have an average saving equal to 1.30%, in 709s as the mean solution time. For S20, Model1 and Model2 can reach 0 and 7 optimal results within the time limit. Their average savings are 1.42% and 1.58%, whereas the mean UB is 1.66%. The average savings of S50 instances by Model1 and Model2 are 2.26% and 2.45% without any optimal outcomes but UB S50 is 2.68%.

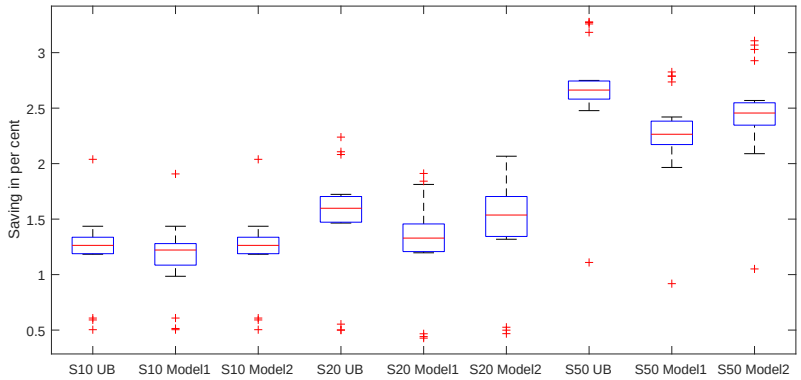


FIGURE 4.7: Saving for instances with 10, 20 and 50 trucks on the Grid10 network

For Grid30, Figures 4.9 and 4.10 present the results and Table 4.5 provides their summary. Since we are confronted with a more complicated road graph, less optimal outcomes and longer computational times are expected. Much larger difference is observed between the models' abilities. S10 instances are solved by Model1 returning only 5 optimal solutions, while Model2 can still find the optimal solutions of all. The average saving and execution time are 1.09% and 1782s for Model1, and 1.24% and 754s for Model2. Considering the S20 results, the optimal outcomes of the two models are no and only 6, respectively. The corresponding average saving are 1.28% for Model1 and 1.42% for Model2. The mean solution

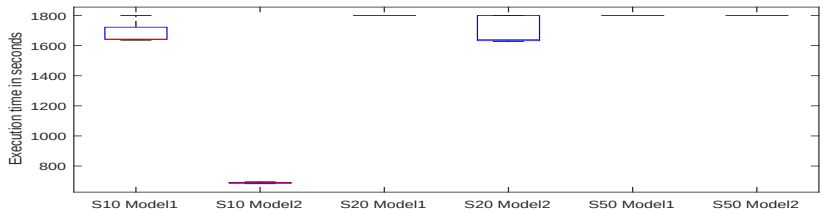


FIGURE 4.8: Execution time of solving for instances with 10, 20 and 50 trucks on the Grid10 network

TABLE 4.4: The number of obtained optimal solutions, average savings and solution times for the Grid10 network with Model1 and Model2

Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	10	20	1.15	1.30	1.30	1733	709
S20	0	7	1.42	1.58	1.66	1800	1751
S50	0	0	2.26	2.45	2.68	1800	1800

time is 1778s. The average saving of UB, Model1 and Model2 are 2.56%, 2.17% and 2.36% for S50 samples.

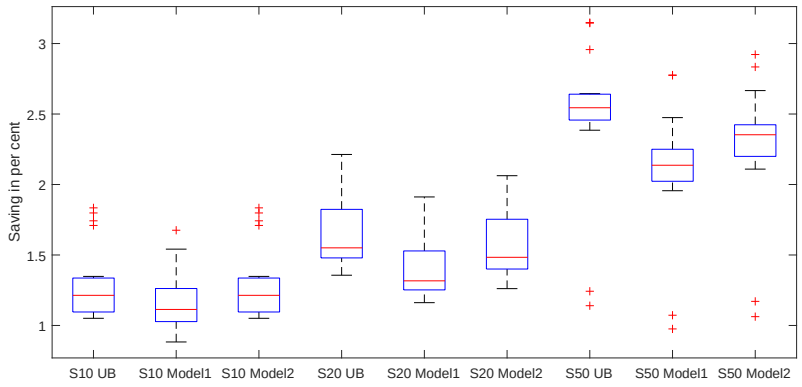


FIGURE 4.9: Saving for instances with 10, 20 and 50 trucks on the Grid30 network

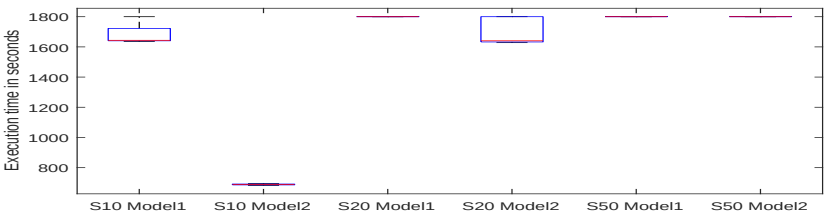


FIGURE 4.10: Execution time of solving for instances with 10, 20 and 50 trucks on the Grid30 network

Figures 4.11 and 4.12 are dedicated to the results on Grid50 network. Similarly, the means of results are shown in Table 4.6. It is the most complex graph used in this thesis to solve FEP instances on. As for S10 instances, we observe only one optimal solution obtained by Model1 (mean=1.15%), whereas Model2 can still find

TABLE 4.5: The number of obtained optimal solutions, average savings and solution times for the Grid30 network with Model1 and Model2

Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	5	20	1.09	1.24	1.24	1782	754
S20	0	6	1.28	1.42	1.51	1800	1778
S50	0	0	2.17	2.36	2.56	1800	1800

all the optimal solutions (mean=1.21%) in 832s on average. Model1 cannot find the optimal solution of any instance of S20 within the solver time limit and its average final saving is 1.13%. Contrarily, Model2 reaches 3 optimal savings and its overall average saving is 1.24% but the UB average is 1.33%, which shows a non-trivial difference. For S50, like all the previous networks, no optimal result is obtained through the exact solving process of any of the models. The best savings of Model1 and Model2 are 2.09% and 2.26%, respectively, which are considerably lower than the average UB equal to 2.48%.

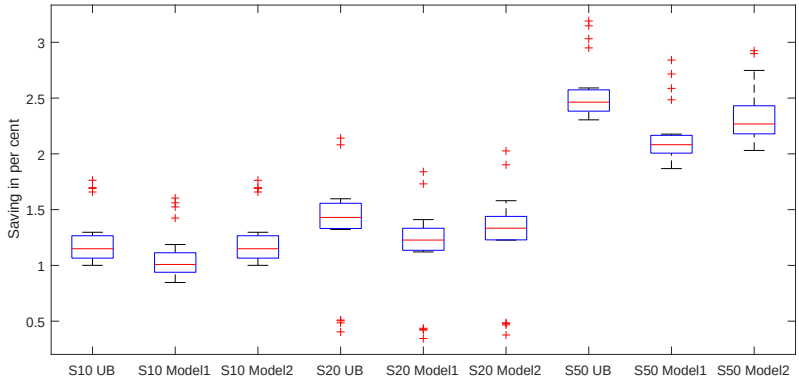


FIGURE 4.11: Saving for instances with 10, 20 and 50 trucks on the Grid50 network

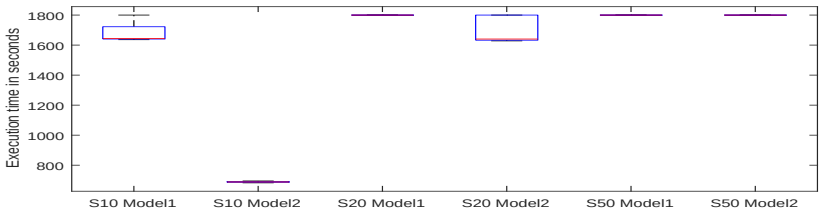


FIGURE 4.12: Execution time of solving for instances with 10, 20 and 50 trucks on the Grid50 network

As expected, due to more feasible routes available for each vehicle resulting in more platooning opportunities, the saving rates on the grid networks are higher in comparison with a normal graph having approximately the same number of nodes and edges. However, the solution time increases in such graphs.

Generally, as it is evident from all the figures, the CPLEX solver can more easily solve Model2 rather than Model1. Higher saving percentages near to upper bounds in shorter computational times can be found through Model2 in comparison with Model1. By increasing the problem size, the saving rate is improved, however, the computational time increases and the number of obtainable optimal solutions

TABLE 4.6: The number of obtained optimal solutions, average savings and solution times for the Grid50 network with Model1 and Model2

Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	1	20	1.15	1.21	1.21	1799	832
S20	0	3	1.13	1.24	1.33	1800	1798
S50	0	0	2.09	2.26	2.48	1800	1800

decreases rapidly. This happens also gradually as the road network becomes harder by going from the Chicago to Grid50 network. In terms of S50, as mentioned, none of the 20 instances can be tackled on any of the six networks, so the gaps between the upper bounds and best found solutions are quite large even with Model2.

The proof for the NP-hardness of a very simple version of the platooning problem, with the same destination, single speed, and without any time and distance constraints for vehicles, is given in *Larsson et al. 2015* [54]. This is based on matching with the simplest set covering problem, which has already been proved to be NP-hard in *Karp (1972)* [45].

Hence some methods are needed to reduce the complexity and increase the tractable scale of the problem by the exact solver. This is investigated in the following of this chapter. The savings and times show the superiority of Model2 over Model1. Nonetheless, some statistical tests about the performance of these models and the decomplexified version of them should be performed, which is our last subject in the chapter.

4.2 Complexity Reduction

Since the inability of the exact optimiser to tackle the instances including 50 vehicles and more is observed, and it has been proved that the problem is NP-hard, some strategies should be applied to reduce the complexity. It is sought to increase the maximum scale that can be solved by GAMS/CPLEX, however, it should not be expected that these strategies make the problem solvable in much larger sizes. This is because the resulted models are still complex.

The constraints related to the departure and arrival time of vehicles, maximum detour as well as multiple speed profiles are specific factors in our FEP problem that by themselves make the solution process more complicated. Nonetheless, these factors can also be intelligently used to reduce the complexity. In this sense, any single vehicle and also the pairwise platooning options as well as the edges and nodes of the graph are analysed according to the vehicle and graph data to pre-assign the optimum values to some variables in the beginning. So they are not included as a variable in the solution process.

Here the possible methods to decrease the problem complexity are explained with the aid of some notations presented in Section 3.2.1 of the previous chapter:

- For each truck if traversing an edge makes respecting the deadline impossible, this edge is not chosen. It means that the fastest arrival time at the beginning of the edge plus the fastest required traversal duration along the edge added to the fastest travelling time from the end of the edge to the destination is larger than the deadline. In this case, zero is assigned to the corresponding decision variable. The fastest means with the highest speed (s_1) on the shortest path for the corresponding connection. This is mathematically expressed as:

$$If \quad T_e^v + \frac{SP_{O^v i} + l_{ij} + SP_{jD^v}}{\delta_{s_1}} > T_{max}^v \Rightarrow x_{ijs}^v = 0$$
$$v \in V; i, j \in G, e_{ij} \in E, s \in S \quad (4.1)$$

- For each truck, if the length of the shortest route which includes an edge is longer than the shortest path plus the maximum allowable detour of the truck, then the decision variable corresponding to the assignment of this edge to the truck is set to zero.

$$If \quad SP_{O^v i} + l_{ij} + SP_{jD^v} > SP_{O^v D^v} + MD^v \Rightarrow x_{ijs}^v = 0$$
$$v \in V; i, j \in G, e_{ij} \in E; s \in S \quad (4.2)$$

- If platooning of two trucks on a specific edge makes respecting the deadline of any of them impossible, then the corresponding decision variable in the two models must be set to zero.

$$If \quad \max\{T_e^v + \frac{SP_{O^v i}}{\delta_{s_1}}, T_e^w + \frac{SP_{O^w i}}{\delta_{s_1}}\} + \frac{l_{ij}}{\delta_{s_1}} > \min\{T_{max}^v - \frac{SP_{jD^v}}{\delta_{s_1}}, T_{max}^w - \frac{SP_{jD^w}}{\delta_{s_1}}\}$$
$$\Rightarrow \quad p_{ij}^{vw} = 0 \quad in\ Model1, \quad f_{ij}^{wv} = 0 \quad in\ Model2$$
$$v, w \in V; i, j \in G, e_{ij} \in E \quad (4.3)$$

-If a node is by only one edge connected to the rest of the network and it is neither the origin nor the destination of any truck, this node and its edge are eliminated from the model.

The corresponding variables are pre-assigned in GAMS based on the above complexity reductions and the rest of variables are determined in the solution process by the CPLEX optimiser. It is expected that by these methods, which provide new versions of Model1 and Model2, called henceforth decomplexified models, larger problem instances can be tackled. In the next section, we give the results of these decomplexified models.

4.3 Results of Decomplexified Models

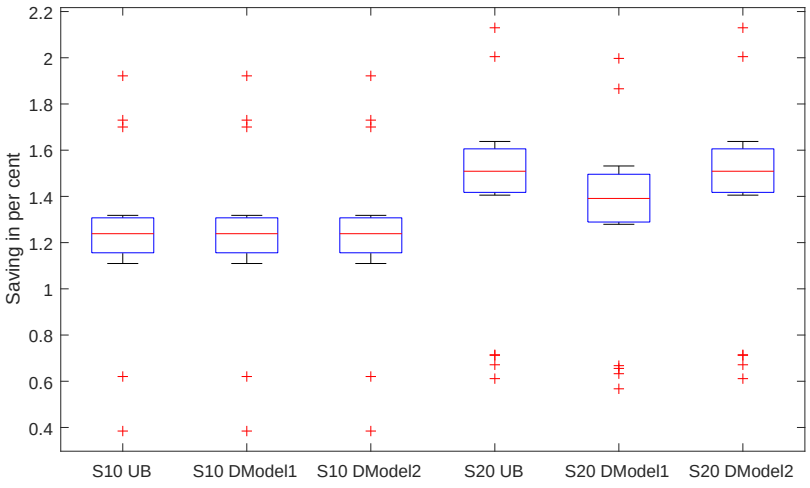
After applying the decomplexifying methods, i.e. pre-assignments of values to some variables in GAMS, the two models, here known as DModel1 and DModel2, are solved agian with the same instances of Section 4.1. The box plots are shown for saving percentages and execution times resulted from solving these decomplexified models.

Since the same instances of the previous section are used, the upper bounds of savings are exactly the same as those of Section 4.1. However, to show the comparison of upper bounds with the savings obtained by DModels, the UB box plots are shown again here. They appear also in the rest of this thesis whenever these instances are solved by any other method.

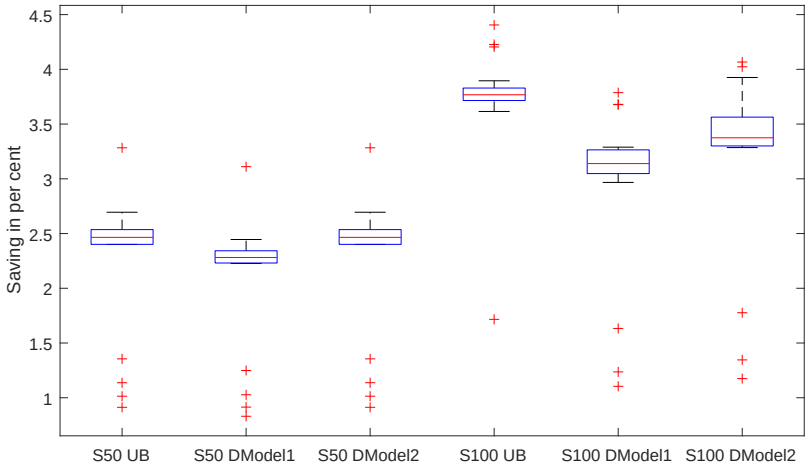
Figures 4.13 to 4.24 illustrate the box plots of savings and elapsed execution times for the six networks. As our new results indicate, by applying our methods of reducing the problem complexity, i.e. using DModels, the maximum exactly (by the exact solver) solvable size within our time limit can be increased to S50. Hence, in this part, our figures showing the achieved savings consist of two sub-figures because they include also the results of S100 which is the smallest unsolvable size thorough any of the DModels.

In the following, some complementary information about the number of attained optimal solutions as well as the means of savings and execution times on each of the six graphs are presented and the performance of the DModels are reviewed.

Figures 4.13 and 4.14 display the results on the Chicago network, and their averages are given in Table 4.7. For S10 , here DModel1 is enabled to solve all the instances to optimality (Model1 could achieve only 15 of them) on average in 1624s. Thus, its average saving is the same as the one of UB and Model2 presented in Section 4.1. DModel2 also returns the optimal savings of all instances, however, its average solution time is reduced to 255s, which is less than half of the one of Model2. For S20, DModel1 is still unable to end up with any optimal results (average saving=1.31%), however DModel2 is now able to reach all of the optimal solutions on average in 660s (Model2 returned only 13 optimal solutions). Examining S50, DModel1 cannot reach the optimality in any case but DModel2 does for all the cases. The average saving obtained through DModel1 is 2.09%, however, this is 2.26% obtained in a mean time of 1609s by DModel2. From S100, challenges of the exact solver with DModel2 begin crucially, therefore, it cannot locate any optimal solution. The average saving of DModel2 is 3.27% but the average upper bound indicate a larger rate of 3.53%.



(A) Instances with 10 and 20 trucks



(B) Instances with 50 and 100 trucks

FIGURE 4.13: Saving on the Chicago network

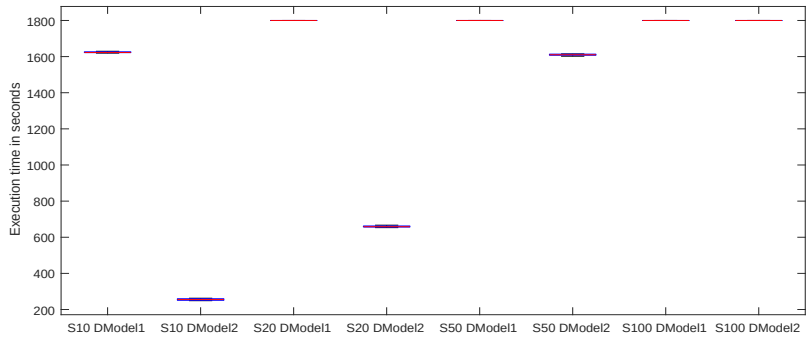


FIGURE 4.14: Execution time of solving for instances with 10, 20, 50 and 100 trucks on the Chicago network

TABLE 4.7: The number of obtained optimal solutions, average savings and solution times for the Chicago network with DModel1 and DModel2

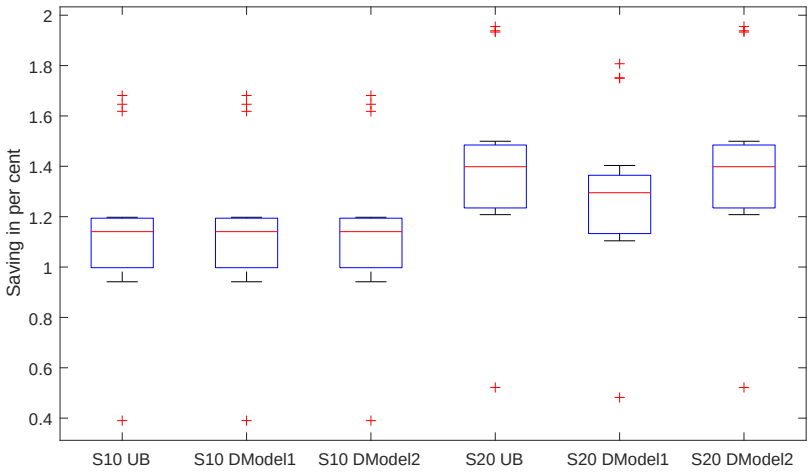
Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	15	20	1.21	1.24	1.24	1680	690
S20	0	13	1.21	1.37	1.41	1800	1683
S50	0	0	1.93	2.09	2.26	1800	1800

Figures 4.15 and 4.16 present the performance of the DModels on the German network. The results' summary is shown in Table 4.8. For S10, both of the models can obtain the complete set of optimal results (though Model1 could found only 13 of them). There is indeed an obvious difference in the solution times of the DModels compared with their initial versions (Model1 and Model2). The average time of DModel1 is 1638s, while it is 268s for DModel2, that is 0.38 of the average solution time of Model2. For S20, DModel1 cannot solve any of the instances to optimality within 30 min, contrarily, DModel2 is now able to give all the optimal results (Model2 found only half of them) on average in 673s. For S50 examples, DModel2 provides optimal savings for all the cases in an average time of 1618s, however, DModel1 cannot do it for any. For S100, the mean saving of DModel2 is equal to 3.20% without any optimal outcome, though, the S100 UB mean is 3.47%.

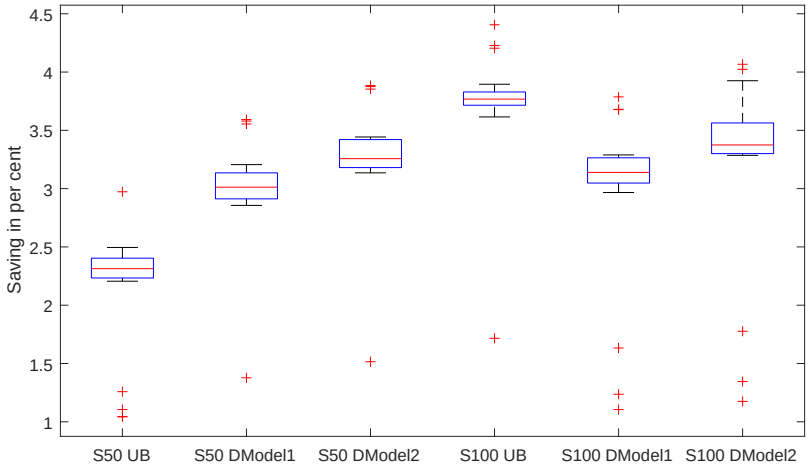
TABLE 4.8: The number of obtained optimal solutions, average savings and solution times for the German network with DModel1 and DModel2

Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	13	20	1.11	1.14	1.14	1704	693
S20	0	10	1.19	1.35	1.40	1800	1726
S50	0	0	1.83	1.95	2.13	1800	1800

Figures 4.17 and 4.18 show the box plots of saving and time on the Swedish network. Their means are presented in Table 4.9. In terms of S10, DModel1 and DModel2 can achieve 18 (Nr. of optimal results by Model1=12) and all of the optimal savings in average times of 1667s and 290s (0.39 of the one of Model2), respectively. Considering S20, DModel1 cannot return any optimality (average saving=1.04%) but DModel2 ends up with the optimal solution of all instances (against only 9 of Model2) in a mean time of 691s. For S50, while DModel2 finds all the 20 optimal solutions in a mean time of 1641s, DModel1 is again completely unable. For S100



(A) Instances with 10 and 20 trucks



(B) Instances with 50 and 100 trucks

FIGURE 4.15: Saving on the German network

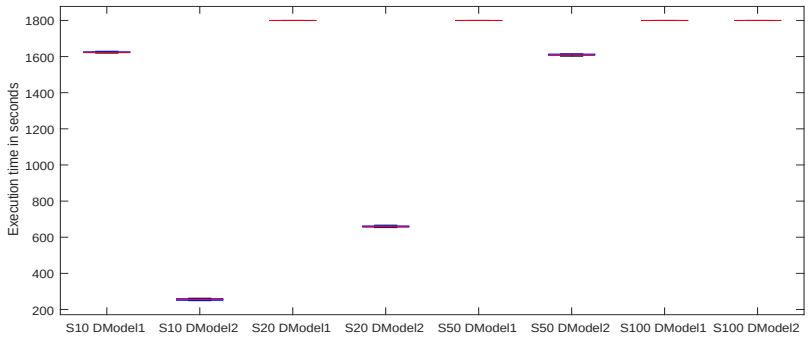
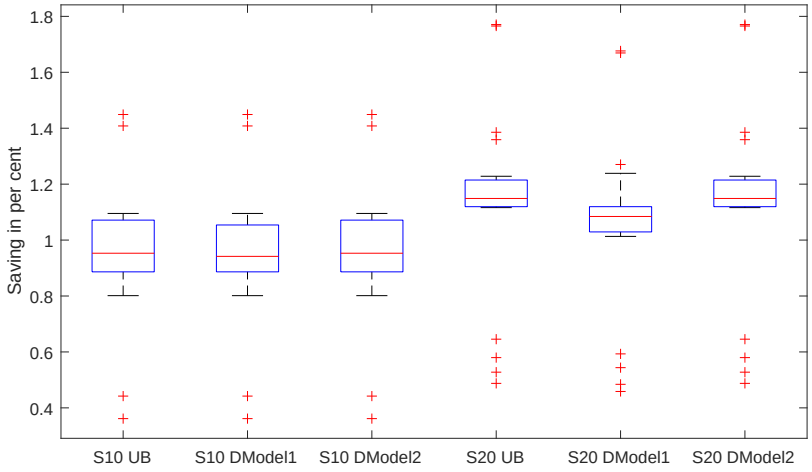
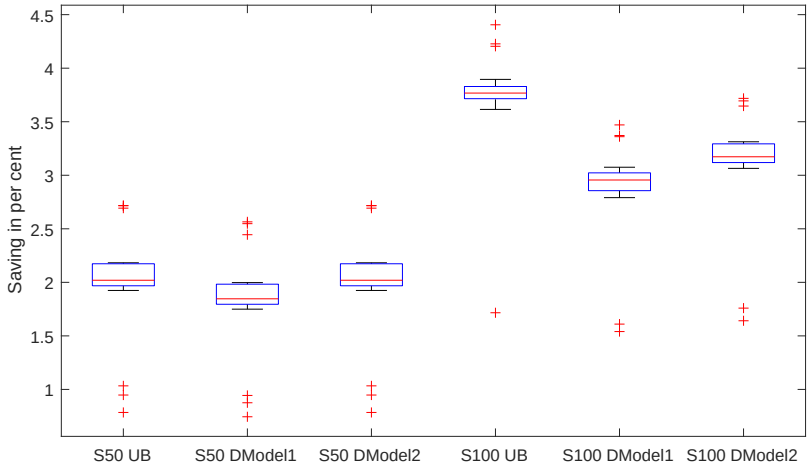


FIGURE 4.16: Execution time of solving for instances with 10, 20, 50 and 100 trucks on the German network

instances, DModel2 cannot find any optimal saving and its average saving is 3.11%, which considerably differs from the average UB equal to 3.36%.



(A) Instances with 10 and 20 trucks



(B) Instances with 50 and 100 trucks

FIGURE 4.17: Saving on the Swedish network

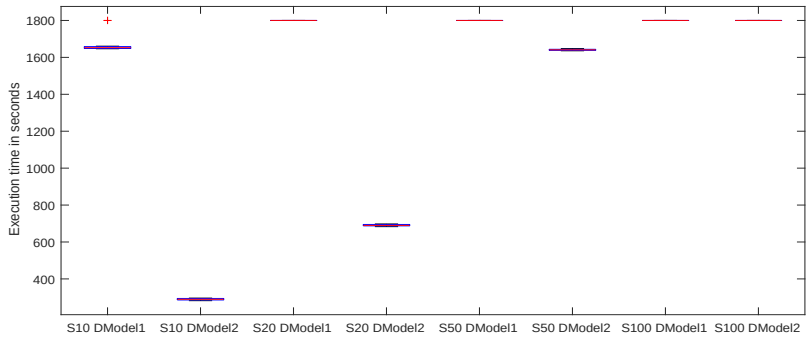


FIGURE 4.18: Execution time of solving for instances with 10, 20, 50 and 100 trucks on the Swedish network

Figures 4.19 and 4.20 include the box plots of saving and time on the Grid10 network, and Table 4.10 gives the number of optimal solutions and the averages. For

TABLE 4.9: The number of obtained optimal solutions, average savings and solution times for the Swedish network with DModel1 and DModel2

Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	12	20	0.91	0.95	0.95	1720	716
S20	0	9	0.96	1.07	1.12	1800	1739
S50	0	0	1.67	1.82	1.97	1800	1800

S10, DModel1 results in 16 optimal savings (Nr. of optimal results by Model1=10) in 1648s on average, whereas DModel2 finds all the optimal solutions in an average time of 291s (the mean time by Model2=709s). For S20, DModel1 cannot finds any optimal solution but DModel2 does it for all instances (Model2 can achieve only 7 of them) on average in 697s. Solving S50 instances, DModel1 gives an average saving of 2.45% for its non-optimal outcomes, however, DModel2 gives a 2.65% average saving in a mean time of 1668s, which is corresponding to 18 optimal solutions. The S50 UB is 2.68%. For S100, average DModel2 saving is 3.29% but the mean S100 UB is 3.62%.

TABLE 4.10: The number of obtained optimal solutions, average savings and solution times for the Grid10 network with DModel1 and DModel2

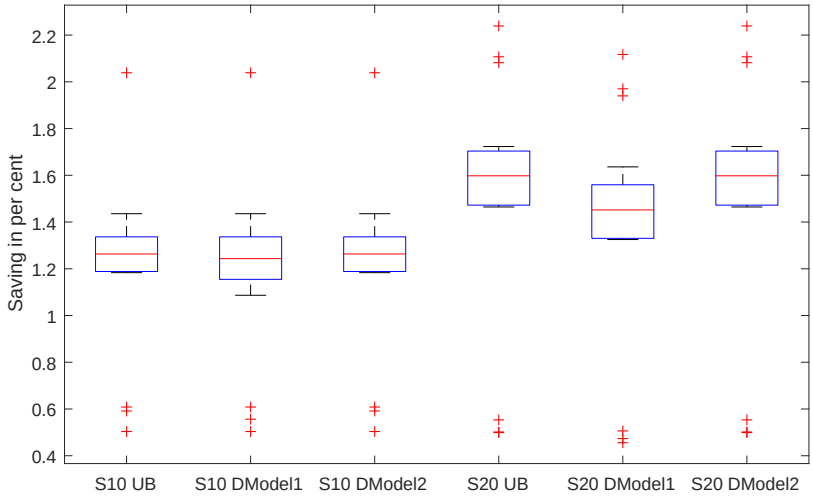
Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	10	20	1.15	1.30	1.30	1733	709
S20	0	7	1.42	1.58	1.66	1800	1751
S50	0	0	2.26	2.45	2.68	1800	1800

Figures 4.21 and 4.22 present the results on Grid30 network. Their summary is shown in Table 4.11. For S10, DModel1 returns 8 optimal solutions (Model1 gives only 5) but DModel2 gives all the optimal solutions in 347s (Model2 execution time= 754s). For S20, DModel2 can solve all the 20 instances to optimality (Model2 can only 6 of them) in a mean execution time of 735s, though, DModel1 cannot any. For S50, DModel2 locates 18 optimal solutions in a mean time of 1706s, however, for S100, it cannot find any and provides an average saving of 3.18%. Nonetheless, the average UB is 3.45%.

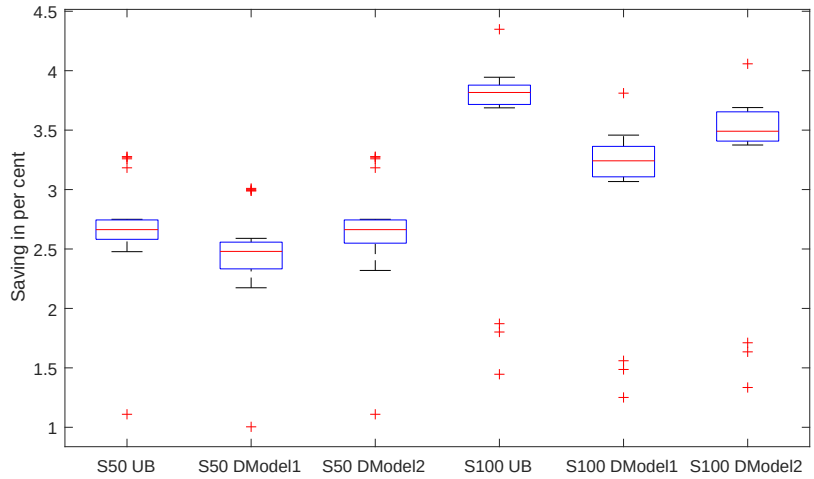
TABLE 4.11: The number of obtained optimal solutions, average savings and solution times for the Grid30 network with DModel1 and DModel2

Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	5	20	1.09	1.24	1.24	1782	754
S20	0	6	1.28	1.42	1.51	1800	1778
S50	0	0	2.17	2.36	2.56	1800	1800

Finally, Figures 4.23, 4.24 and Table 4.12 are dedicated to Grid50 network. For S10, DModel1 can only solve 6 instances to optimality (Nr. of optimally solved instances by Model1= 1) but DModel2 finds all optimal savings averagely in 410s (less than the half of the average time of Model2). For S20, DModel2 obtains all the optimal savings (Model2 can provide only 3 of them) in 804s, however, DModel1 cannot find any. For S50, 17 optimal solutions are found through solving DModel2 in an average time of 1774s. However, for S100 instances, like with the previous



(A) Instances with 10 and 20 trucks



(B) Instances with 50 and 100 trucks

FIGURE 4.19: Saving on the Grid10 network

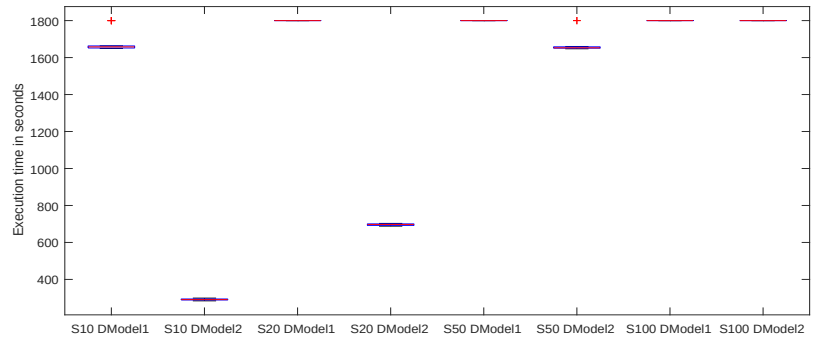
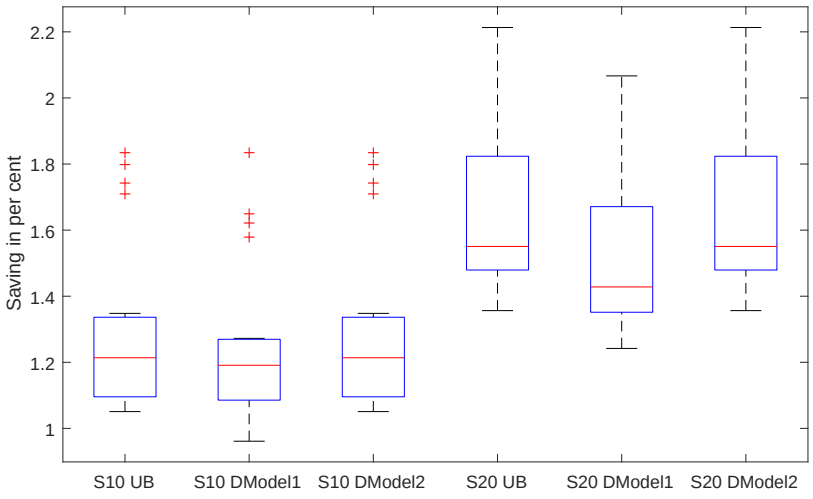
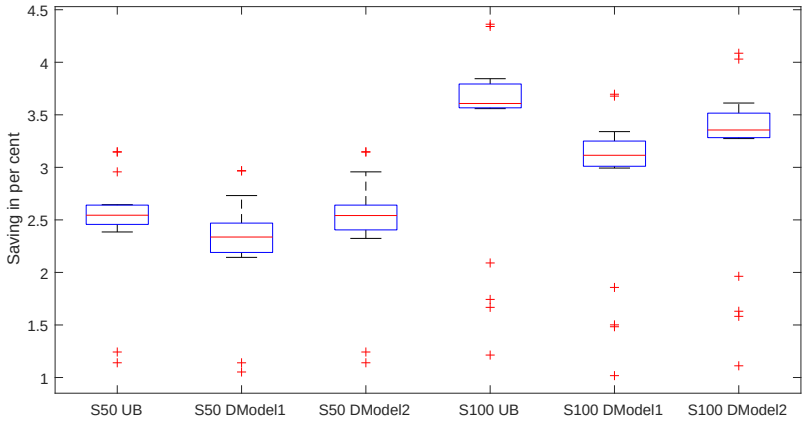


FIGURE 4.20: Execution time of solving for instances with 10, 20, 50 and 100 trucks on the Grid10 network



(A) Instances with 10 and 20 trucks



(B) Instances with 50 and 100 trucks

FIGURE 4.21: Saving on the Grid30 network

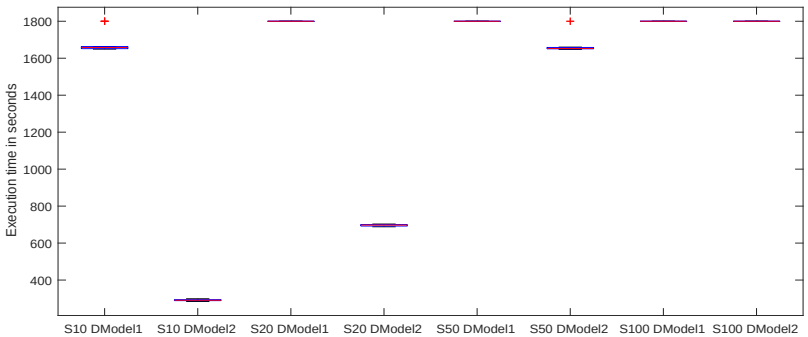
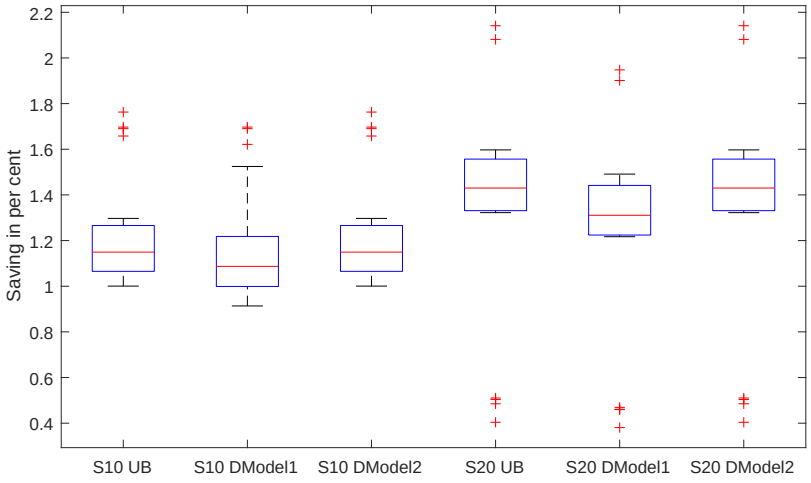


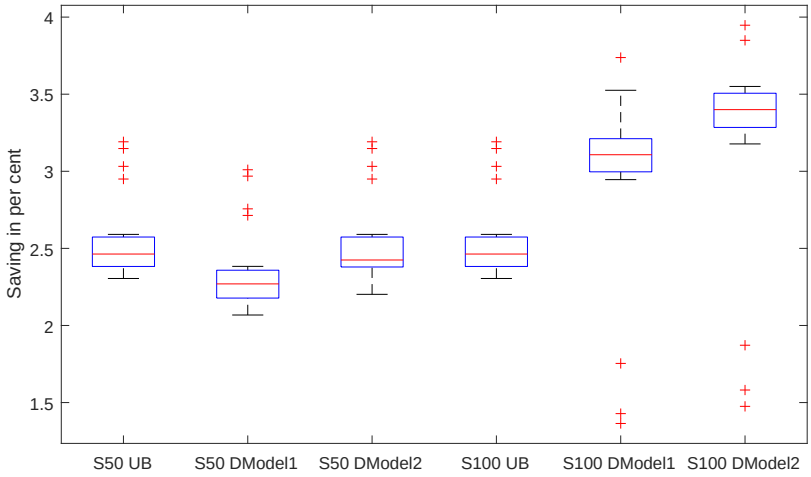
FIGURE 4.22: Execution time of solving for instances with 10, 20, 50 and 100 trucks on the Grid30 network

networks, DModel2 is not able to return any optimal solution within our 30 min time. Its average saving is equal to 3.11% against the S100 UB that is 3.34%.

Generally, the obtained results show that by the DModels, we can increase the



(A) Instances with 10 and 20 trucks



(B) Instances with 50 and 100 trucks

FIGURE 4.23: Saving on the Grid50 network

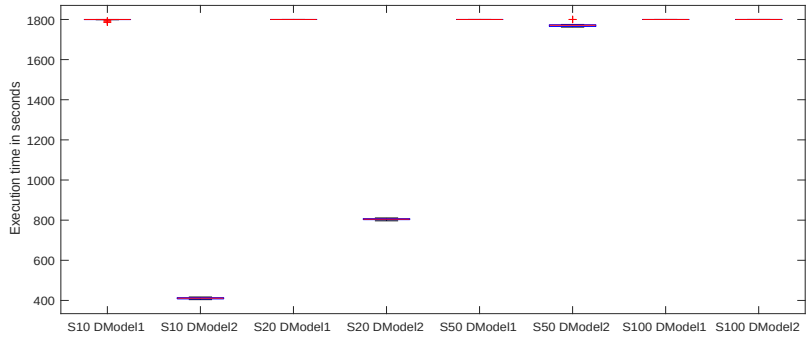


FIGURE 4.24: Execution time of solving for instances with 10, 20, 50 and 100 trucks on the Grid50 network

exactly solvable size, and considerably decrease the solution time. However, S50,

TABLE 4.12: The number of obtained optimal solutions, average savings and solution times for the Grid50 network with DModel1 and DModel2

Problem Size	Nr. of Optimal Solutions		Avr. Saving (%)			Avr. Time (s)	
	Model1	Model2	Model1	Model2	UB	Model1	Model2
S10	1	20	1.15	1.21	1.21	1800	832
S20	0	3	1.13	1.24	1.33	1800	1799
S50	0	0	2.09	2.26	2.48	1800	1800

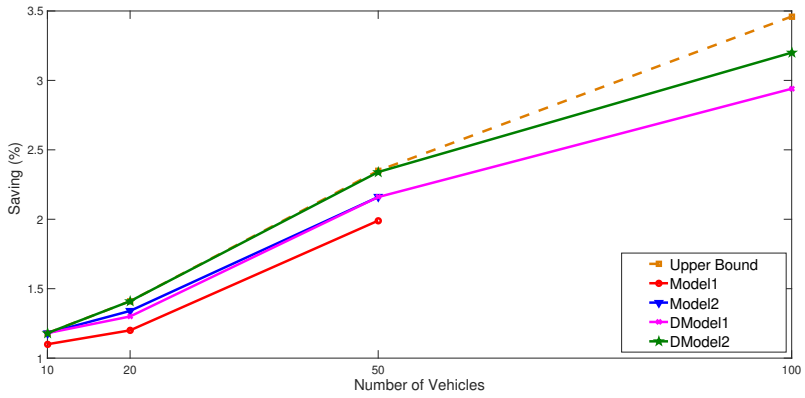


FIGURE 4.25: The overall average of the savings provided by the 4 models regarding all the 6 road networks

for which most of the instances are solved to optimality with DModel2, is still much smaller than the real size of an FEP problem. Therefore, it should be sought for other alternative methods which are able to find good savings for larger sizes of this problem in real time.

At the end of this section, the overall average of the saving and time of the models regarding all the road network graphs are illustrated in Figure 4.25 and 4.26 respectively. As it is evident, the overall average savings provided by DModel2 is above and its solution times are under the other three models. This indicate the dominance of this model over the others.

4.4 Statistical Tests

The results of solving the FEP instances of different sizes on the 6 networks indicated the performance of Model1, Model2, DModel1 and DModel2 by the corresponding savings and solution times. However, in this section, it is aimed at presenting a statistical comparison among the performance of all the 4 Models in terms of both saving and solution time.

There are several statistical tests to compare multiple methods together. Some of them require the normality of results within any group, though, in most of the cases this cannot be proved with a high confidence level. Hence, here a very appropriate non-parametric method called the Friedman test with the Bergmann-Hommel post hoc procedure, see *Bergmann and Hummels (1988)* [8], is applied. This is according to *Derrac et al. (2011)* [22] one of the best approaches for such multiple comparisons and do not need any normality proof of the data. This test is used in the rest of this thesis whenever it is aimed at making a comparison between multiple approaches.

The results obtained on all the 6 networks are grouped based on the number of vehicles (problem size). So the 20 results with the same number of vehicles, i.e. S10, S20, S50 and S100, are accumulated from all networks for each of the 4 Models,

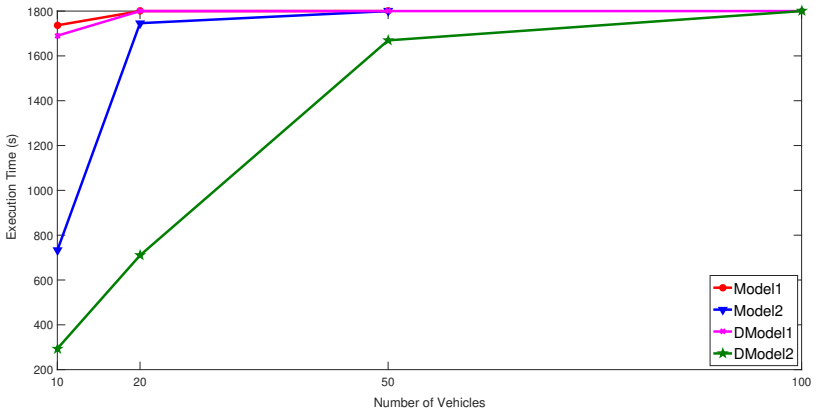


FIGURE 4.26: The overall average of the solution times of the 4 models regarding all the 6 road networks

which results in 120 data for saving and time in each group. This is a proper number of data for our statistical comparisons. For each problem size, the results of all models are compared in pairs to find if there is a significant difference between them in terms of both savings and execution times. We implement these tests in *R* language [75] with the *SCMAMP* [1] package [15].

Table 4.13 contains the p-values of these statistical comparisons between any pair of the models for each size of the vehicle set in terms of savings. As it is evident, based on the obtained p-values, which are mostly near 0, there is a significant difference between the savings of all the pairs in all the sizes except for Model2 vs. DModel1 for S50 and S100, as well as Model2 vs. DModel2 for S10, that p-values do not show any significant difference.

Comparison \ Size	S10	S20	S50	S100
Model1 vs. Model2	0.0000e+00	0.0000e+00	0.0000	0.0000e+00
Model1 vs. DModol1	2.3651e-10	3.1086e-15	0.0000	0.0000e+00
Model1 vs. DModel2	0.0000e+00	0.0000e+00	0	0
Model2 vs. DModel1	2.0456e-07	1.8511e-06	0.9885	0.9915
Model2 vs. DModel2	1.0000e+00	3.4974e-10	0	0.0000
DModel1 vs. DModel2	2.04561e-07	0.0000e+00	0.0000	0.0000

TABLE 4.13: The p-values of the pairwise statistical comparisons of Model1, Model2, DModel1 and DModel2 savings by the Friedman test with Bergmann-Hommel post-hoc procedure

Generally, regarding the savings of the Models shown in Sections 4.1 and 4.3, and the pairwise *Friedman/Bergmann-Hommel* tests of this section, the Models can be sorted in terms of saving as DModel2 $\succ$ Model2 $\succ$ DModel1 $\succ$ Model1 in almost all of the sizes.

Similarly, the p-values of comparing the execution times are shown in Table 4.14. As none of the Models can solve S100, the tests are only done for S10, S20 and S50. It can be perceived from the very low p-values that a statistically significant difference exists between the time of all pairs in solving S10 and S20 instances except Model1 vs. DModel1 for S20. However, in terms of S50, the differences are only evident in comparisons between DModel2 and other Models. Since a time limit of 30 min applies, if solution process of an instance through any model is not

[1] Statistical Comparison of Multiple Algorithms in Multiple Problems

finished within this limit, it is terminated and 1800s is recorded as its execution time. Therefore, the time of two models may seem as being the same (1800s) because they both are unable to solve instances of a size to optimality and stop before the end. However, in fact, they are not the same if we continue the solution process. Model1 vs. DModel1 for S20 and S50 are examples for this. Generally, based on the results and tests, the models from the shorter (better) to longer required solution times are: DModel2 $\succ$ Model2 $\succ$ DModel1 $\succ$ Model1.

Comparison \ Size	S10	S20	S50
Model1 vs. Model2	0.0000e+00	2.6096e-09	1
Model1 vs. DModol1	2.0966e-05	1.0000e+00	1
Model1 vs. DModel2	0.0000e+00	0	0
Model2 vs. DModel1	2.2204e-15	2.6096e-09	1
Model2 vs. DModel2	4.1335e-11	0	0
DMode1 vs. DModel2	0.0000e+00	0.0000e+00	0

TABLE 4.14: The p-values of the pairwise statistical comparisons of Model1, Model2, DModel1 and DModel2 times by the Friedman test with Bergmann-Hommel post-hoc procedure

4.5 Summary

In this chapter, we started from presenting the results of solving the two Models introduced in the previous chapter in GAMS by the CPLEX solver. The performances with 20 instances on our six networks were presented. The savings and required solution times were shown by box plots for each size and model. So the range of the results could be compared together. Due to the weakness of the solver in dealing with problems including 50 trucks and more, some decomplexifying strategies are proposed. It was shown that these strategies enable the second model to solve problems with 50 trucks and reduce the execution times for smaller sizes with both the models.

Even with these complexity reduction methods, the maximum solvable size cannot be bigger than the ones with 50 vehicles. However, in the reality, much larger FEP instances must be solved. Hence, in the next chapters, other alternative solution methodologies, which allow us to deal with the problem in larger scales, are proposed.

Chapter 5

Heuristic Methods

In the previous chapters, two mathematical models for our fuel-efficient HDV platooning problem are presented and some generated samples of different sizes are solved through these models coded in GAMS by the CPLEX solver. Since it is observed that even the powerful employed solver is unable of tackling the problem as the number of vehicles increases to 100, we strongly require some other solution methodologies which can stay applicable and efficient by larger samples. Hence, in this chapter, heuristic algorithms that have been specifically designed to solve our problem are introduced and applied. Three methodologies are presented, namely: Best Pair, Hub and Global Planning heuristic. The two first have been adapted from the literature and modified to be applicable to our version of FEP problem, whereas the third is completely original of this work. To enhance the performance of our heuristics and obtain better results, a local search strategy is added to the end of them. This chapter is structured as: the three first sections, 5.1 to 5.3, explain the heuristic approaches. The attached Local Search approach is explained in Section 5.4 and its benefit is examined in Section 5.6. The results of the methods are presented and statistically compared in Sections 5.5 and 5.7, respectively. At last, a summary of this chapter is drawn in Section 5.8.

5.1 Best Pair

Larsson et al. (2015) [54] develop an algorithm, called the Best Pair heuristic, for the unlimited platooning problem, based on the heuristic of *Larson et al. (2013)* [53]. We use the concept of this algorithm and make a heuristic approach with the same name to solve our version of FEP, which has time and distance constraints, and multiple speeds for vehicles.

This Best Pair heuristic algorithm iteratively chooses the current best pair of platoons to merge together by searching over all pairs and their best merging and splitting nodes. The major difference in our Best Pair heuristic is that searching for the best merging and splitting nodes is done only among feasible nodes which do not violate the constraints of our problem if they are chosen. In the following, our Best Pair heuristic algorithm is explained based on the concepts and notations presented in Subsection 3.2.1 of Chapter 3. According to the steps, the explanations are given in the three following subsections.

5.1.1 Feasible nodes and routes

Those nodes are called feasible that if any of the two vehicles of a pair passes through them, it can then respect its deadline and also maximum allowable detour. In order to find such nodes, an algorithm is firstly applied to find all paths for each vehicle v from its origin O^v to destination D^v which can be traversed by the deadline T^v_{max} and are not longer than the shortest path $SP_{O^v D^v}$ plus the maximum detour MD^v. These routes are also called feasible. If a node is within the feasible routes of both vehicles, it is a feasible node for the pair.

The feasible routes between two nodes are found by a search tree. First of all, the shortest path is found by Dijkstra's algorithm (*Cormen et al., 2001*[17]) and its

length, $SP_{O^v D^v}$, is calculated accordingly. Then, this tree starts from the origin and branches to all the connected nodes. For each branch, the distance from the origin is calculated. If this value is larger than the minimum of $(T^v_{max} - T^v_e) \times \delta_{s_1}$ and $SP_{O^v D^v} + MD^v$, which means that the route becomes infeasible, branching from that point is terminated. The first term is the longest route length that the vehicle can travel on and still respect its deadline calculated based on driving at the maximum speed (s_1). The second term is the longest route length allowed by the maximum detour. Branching is continued until either the destination node is reached or the route is bounded due to infeasibility. It is not allowed to branch into an already visited node. In the end of this procedure, we can obtain all the feasible routes for a vehicle to reach its destination from the staring point. This can be explained as pseudocode in Algorithm 1.

Algorithm 1: Tree search procedure to find all feasible routes between two nodes

Data: O^v, D^v, T^v_e, T^v_{max} and the network
Result: All feasible routes between the origin and destination
- Find the shortest path between O^v and D^v by Dijkstra's algorithm and calculate $SP_{O^v D^v}$.
- Start from the origin.
while *Any branching is possible* **do**
    - Branch over any open node to the unvisited connected nodes and calculate the distance from the origin.
    - Bound any branch (route) if the distance is longer than $min((T^v_{max} - T^v_e) \times \delta_{s_1}, SP_{O^v D^v} + MD^v)$.
    - Close any node if the destination is reached.
end
- For any unbounded branch that has reached the destination, construct the feasible route by writing the nodes from the top to down.

The execution of Algorithm 1 for a simple example and the corresponding search tree are shown together in Figure 5.1. It is assumed that there is a vehicle going from node 1 to 5. Its time constraints are $T^v_e = 0$ and $T^v_{max} = 5$. The obtained shortest path length is 2 corresponding the route 1-3-5 (shown by "SP" underneath in the tree). Therefore, according to $MD^v = 0.09 \times SP_{O^v, D^v}$, the limit is set to $min(5 \times 1.36 = 6.8, 2 + (0.09 \times 2) = 2.18) = 2.18$, and wherever this amount is exceeded in the tree, branching must be terminated. In our case, this limit is violated before arriving the destination in one case (1-4-3) and after it in the two cases (1-2-3-4-5 and 1-2-3-5), which are corresponding to infeasible routes. The destinations or final threads in the tree are shown in green for feasible routes and in red for infeasible ones or when the branching is stopped. Based on this search tree, all the obtained feasible routes from node 1 to 5 and their lengths are listed in Table 5.1.

TABLE 5.1: Feasible routes from node 1 to 5 for the example of Figure 5.1

Feasible paths between nodes 1 and 5	Length
1-2-5	2.18
1-3-2-5	2.12
1-3-4-5	2.25
1-3-5	2
1-4-5	2.10

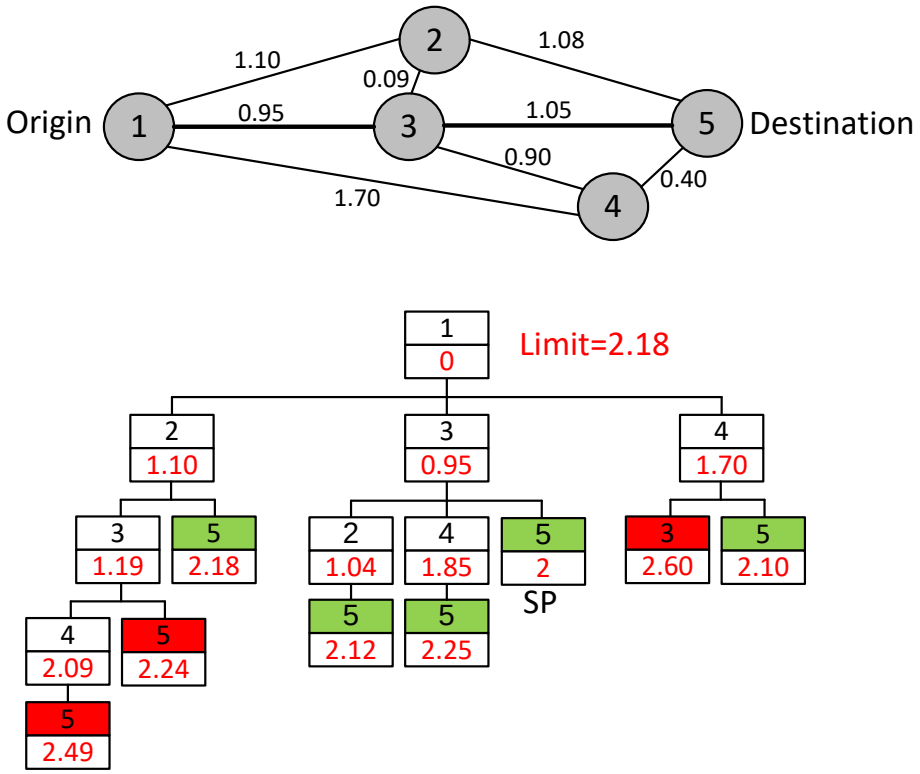


FIGURE 5.1: A simple graph and search tree for finding all the feasible routes from node 1 to 5, i.e execution of Algorithm 1, for a vehicle with $T_e^v = 0$ and $T_{max}^v = 5$.

5.1.2 Scheduling and speed adjustment

In each iteration, the highest possible saving of platooning among all pairs of vehicles based on the best merging and splitting node should be found.

However, calculation of any platooning saving or objective value is not possible without having a complete time schedule and speed adjustment for all vehicles. A time and speed plan for the system that provides a high platooning benefit is derived by a novel scheduling and an embedded deadline violation resolution algorithm. They are significant contributions in this work and used henceforth in all of the heuristics of this chapter and meta-heuristics of the next chapter. By having a complete route, it is given to Algorithm 2 to be converted to a comprehensive platooning plan including time scheduling and speed determination.

This scheduling algorithm initially provides a time plan based on the maximum platooning and driving at the cruise speed by ignoring the deadlines. After this step, all vehicles that their deadlines are violated are found and put in a set called V', consequently, their violation amounts are calculated. To eliminate the violations, the data of vehicles in V' are processed by Algorithm 3 or deadline violation resolution algorithm. This finds the last edge with platooning on the route of vehicles and follows two strategies to reduce the deadline violation: 1. Increasing the speed, and 2. Cancelling the waiting and related platooning. The violation resolution algorithm is a branch and bound approach (B&B) like Algorithm 1 that calculates the benefit which is lost by violation reduction actions. Wherever this loss is more than the minimum amount found so far which leads to respecting the deadline, the corresponding node is bounded and not further investigated. The implementation of this deadline violation resolution algorithm for a simple example is shown in Figure 5.2. After gaining the maximum possible benefit by driving platoons at the cruise speed, Algorithm 2 continues and it is tested whether it is possible to attain

more profit by merging the vehicles that must drive at an identical higher speed on the same edge according to the best scheduling derived so far. At the end of this step, some runs of Algorithm 3 may be needed to correct any new deadline violation. If any vehicle do not take part in any platoon based on the final schedule, it is returned to its shortest path. Finally, the last task is to evaluate the objective function based on the determined time plan and speed adjustment.

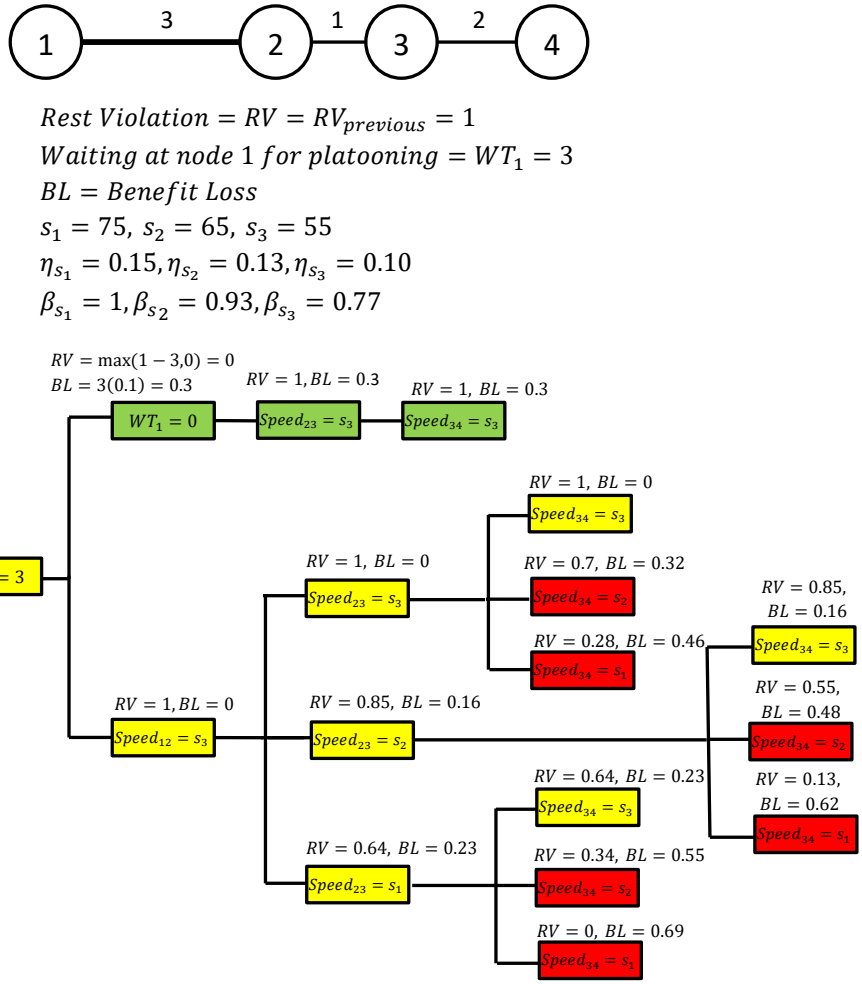


FIGURE 5.2: A simple visual example of implementing Algorithm 3 to make a vehicle respect its deadline. The route is shown above. The platooning happens on edge e_{12} and needs a waiting equal to 3 units of time, i.e. $WT_1 = 3$. The yellow nodes are still open, the bounded nodes are shown in red and the green nodes provide a feasible scheduling.

5.1.3 Procedure of the algorithm

Our Best Pair heuristic algorithm begins with a routing solution in which vehicles drive on their shortest path. Most probably there are multiple shortest paths for some vehicles. In this case, one of the shortest paths is chosen randomly.

In each iteration, the pair that provide the maximum saving is found from the comparison of the results provided by Algorithm 2. We merge this pair and consider the two vehicles as a single one with an origin in the merging and a destination at the splitting point. However, the missions of arriving to the merging point from the origins and reaching the destinations from the splitting point still remain for the

Algorithm 2: Time scheduling algorithm and objective evaluation

Data: The routing of vehicles, vehicles' data, network's data
Result: The best time scheduling and the corresponding objective function

- Ignore the deadlines. Based on driving at the cruise speed, make platoons on each edge with all vehicles which traverse on it at any time. The departing time of each platoon is the arrival time of the last vehicle at the beginning of the edge.
- Find vehicles whose deadline has been violated and put them into a set called V'.
- Schedule any $v \notin V'$ to travel at the cruise speed (s_3) on all the edges of its route. If there is no platooning on an edge, the traversal is scheduled at the fastest time, whereas, if there is platooning on an edge, the vehicle starts the edge traversal at the starting time of the platoon.

while $V' \neq \emptyset$ **do**
  for $v \in V'$ **do**
    - Implement Algorithm 3 for v
    - Eliminate v from V'
  end
  - Make a new platoon on each edge that starts its travel along the edge at the time corresponding to the last traversal.
  - Find all vehicles that their deadline has been violated after making new platoons and put them into V'
end
for *Any speed s_i higher than cruise speed* **do**
  - Find the edges where more than two vehicles are to traverse on at speed s_i
  - Schedule the corresponding vehicles to platoon with speed s_i on the edges found
  - Find all vehicles that their deadline is violated and put them into V'
end
while $V' \neq \emptyset$ **do**
  for $v \in V'$ **do**
    - Implement Algorithm 3 for v
    - Eliminate v from V'
  end
  - Make a new platoon on each edge that starts its travel along the edge at the time corresponding to the last traversal.
  - Find all vehicles that their deadline has been violated after making new platoons and put them into V'
end
if *Any vehicle $v \in V$ does not contribute in any Platooning* **then**
  - Return it to its shortest path
end
- Calculate the objective function based on the maximum possible platooning.

Algorithm 3: Deadline violation resolution algorithm

Data: Vehicle v, its T_e^v, T_{max}^v, its route
Result: An excellent feasible time scheduling and speed adjustment for v

while T_{max}^v *is not respected* **do**
- Find the last platooning waiting.
- Branch over all possible acceleration decisions, i.e. cancelling the waiting (corresponding to a platoon) or increasing the speed on edges up to the next waiting, which has been investigated before.
- In each step of the tree if T_{max}^v is respected by a decision, calculate the sum of benefit losses and continue by branching only over one decision of driving at the cruise speed, thereafter.
- After each step, update the lowest lost found leading to the respect of T_{max}^v.
- Stop branching from (bound) any node if the accumulated loss is more than the lowest amount.

end
- Schedule v according to the time and speed plan corresponding to the least loss found.

two vehicles and are thereafter considered as two separate vehicles. Therefore, after each merging, the two vehicles are converted into five, i.e. two from the staring nodes of the two vehicles to the merging point, a single one from the merging to the splitting point, and at last, two from the splitting node going to the destinations of the two vehicles. The key point is that the merging of the first two and last two vehicles (missions) together is not examined in the next iterations because they are once investigated. Moreover, in the first iteration, the maximum saving of all pairs are calculated and they can be used in the next iterations. So the algorithm becomes faster. This heuristic algorithm proceeds iteration by iteration until no saving can be achieved by merging any pair. Pseudocode for the algorithm is presented as Algorithm 4.

In our version of Best Pair heuristic, we use the complex time and distance constraints in favour of accelerating the algorithm by limiting the number of nodes which are examined for any vehicle pair.

5.2 Hub Heuristic

The idea of the Hub heuristic introduced by *Larsson et al. (2015)* [54] is to drive vehicles to their destination via some nodes called hubs. By this method, the whole problem is broken into several sub-problems which are easier to solve because they can be considered each as a same start platooning problem and can be effectively tackled by the heuristic presented in *Larson et al. (2013)* [53]. This idea is again used to develop our own Hub heuristic which is suitable for the FEP problem of this thesis. The embedded same start platooning algorithm is also modified.

In Hub heuristic, the vehicles are divided into partitions and a hub node is selected for each partition. Hence for each hub, we have the two following sub-problems: 1. Conducting its vehicles from their origins into the hub 2. Conducting the vehicles from the hub into their destinations. The second is an obvious same start platooning problem (SSPP), whereas if we reverse the first sub-problem and find routes for vehicles from the hub into the origins it becomes another SSPP.

For dividing the vehicle set into different subsets (groups), points are given to all feasible nodes of each vehicle. These points are calculated based on the

Algorithm 4: Our Best Pair heuristic

Data: The vehicles set V with O^v, D^v, T_e^v, T_{max}^v and the network graph, i.e. N and E

Result: Routing of vehicles from their origin into destination with good platooning benefits

1 - Initialise a routing with all vehicles driving on their shortest path. If there are multiple shortest paths for a vehicle, one of them is randomly chosen.

2 - Initialise an empty set for vehicles pairs which should not be investigated henceforth, $NI = \emptyset$.

3 **while** *Any saving is possible by merging any pair* **do**

4 - Find the largest saving of all pairs except those in $NI = \emptyset$ based on the best merging and splitting node among feasible nodes of each pair. They are obtained by running Algorithm 1 for each vehicle and extracting those nodes which are within the feasible routes of both vehicles of the pair. Then, Algorithm 2 is applied to each combination of two feasible nodes.

5 - Merge the pair corresponding to maximum saving and convert the vehicle missions into the following five: two from the starting points of the two vehicles to the merging point, one from the merging to the splitting node, and finally, two from the splitting node to the destination of the vehicles.

6 - Add the pair of the first two and last two vehicles (missions) to NI.

7 **end**

vehicles' desirability to pass through that node. So they are inversely proportional to the distance of nodes from the nearest node on the shortest path of the vehicle. Therefore, a set called FN^v is defined including all the feasible nodes of v . The point of a node i for vehicle v named p_i^v is calculated based on its distance from the nearest node on the vehicle's shortest path, spd_i^v, and the largest of these distances among all the feasible nodes, $\max_{j \in FN^v}(spd_j^v)$, as:

$$p_i^v = \begin{cases} \dfrac{\max_{j \in FN^v}(spd_j^v) - spd_i^v}{\max_{j \in FN^v}(spd_j^v)} & i \in FN^v \\ 0 & otherwise \end{cases}$$

The vehicles are partitioned based on the pairwise examination of them with the aid of nodes' points. In this attempt, the multiplication of the points for a pair of vehicles on all nodes are summed up. It is called grouping point and is mathematically expressed for a pair v_1 and v_2 as: $gp_{v_1 v_2} = \sum_{i \in N} p_i^{v_1} \times p_i^{v_2}$. It is equivalent to finding the nodes which are feasible for both and multiplying their points. By having the grouping points of all vehicle pairs, a symmetric matrix of them can be built called GP. For each row of GP or each vehicle, we put it with the vehicle corresponding to the maximum value of the row in one partition or group. If all the grouping points of a row are less than 1, then the corresponding vehicle should not be grouped with any other vehicle and is considered to travel alone on its shortest path. Figure 5.3 shows this GP matrix for 5 vehicles and the final partitioning of them accordingly.

After grouping, a hub node is chosen for each group. For this sake, among the nodes which are feasible for all vehicles of the group, the one with maximum sum of points corresponding to its vehicles is chosen as the group hub.

GP Matrix

	V_1	V_2	V_3	V_4	V_5
V_1	0	7.21	0.98	1.58	5.78
V_2	7.21	0	2.12	0.82	6.25
V_3	0.98	2.12	0	5.43	1.28
V_4	1.58	0.82	5.43	0	1.92
V_5	5.78	6.25	1.28	1.92	0

Group1	V_1, V_2, V_5
Group2	V_3, V_4

FIGURE 5.3: Grouping (partitioning) of vehicles based on the matrix GP of grouping points. The maximum of each row is shown in green.

From then on, regarding each group, the two sub-problems of finding the routes of vehicles from each origin to the hub and from the hub into each destination should be solved. For this sake, Algorithm 5 is applied to each group two times. Once for determining vehicle routes to the hub and once again from the hub to their destination. This algorithm is again based on the pairwise calculation of savings and works like Best Pair heuristic with this difference that we search to find only one node for each pair which is either a merging (for the first problem) or splitting node (for the second problem).

Algorithm 5: Routing for each group (partition)

Data: A set of vehicles with their origins (destinations) and their Hub, the network

Result: Routing of the vehicles with good platooning benefits from (to) their origins (destination) to (from) the hub

1 - Initialise an empty set for vehicles pairs which should not be investigated henceforth, $NI = \emptyset$.

2 **while** *Any platooning saving can be achieved by merging any pair* **do**

3 - Find the largest saving of all pairs except those in $NI = \emptyset$ based on the best merging (splitting) node among feasible nodes of each pair. They are obtained by running Algorithm 1 once for each vehicle and extracting those nodes which are within the feasible routes of both vehicles of the pair. Each saving is calculated by Algorithm 2.

4 - Merge this pair and convert the vehicle missions into the following five: two from the starting points of the two vehicles to the merging point, one from the merging to the splitting node, and finally, two from the splitting node to the destination of the vehicles.

5 - Add the pair of the first two and last two vehicles (missions) to NI.

6 **end**

After dividing vehicles into groups, selecting a hub for each group and directing vehicles into their destination via hubs, we obtain a solution for our FEP problem.

The Pseudocode of the whole Hub heuristic is shown as Algorithm 6.

Algorithm 6: Our Hub heuristic

Data: The vehicles set V with O^v, D^v, T_e^v, T_{max}^v and the network graph, i.e. N and E

Result: Routing of vehicles from their origin into destination with good platooning benefits

1 - Calculate p_i^v for any $v \in V$ and $i \in N$
2 - Calculate $gp_{v_1 v_2}$ for any $v_1, v_2 \in V$ and build the matrix GP
3 - Partition set V based on GP
4 - Find a hub for each partition
5 - Solve all the resulted subproblems by Algorithm 5

5.3 Global Planning

In this section, we introduce a completely novel heuristic approach, which works with platooning plans proposed globally by all nodes of the network. In this method, each node proposes its best platooning plan involving the vehicles, for which it is feasible. The node is considered as the merging point of these vehicles and splitting points are found by Algorithm 5. If a node is not feasible for any vehicle, then no plan is proposed by it. After this step, the node plans are sorted based on their platooning savings. If the saving of two or more plans are the same, then the sum of partial platooning contributions of its involved vehicles in other plans are calculated as the second criterion. Lower is the value of this criterion, more prior is the plan. This is because by implementing such a plan, less platooning opportunities of other plans will be lost. This algorithm starts by implementing the plans based on their order. After implementation of each plan, all the involved vehicles are updated by being divided into three parts of arriving into the merging node (the plans' starting node), the platooning section of the road, and finally arriving into the destination from the splitting nodes. Subsequently, the other plans are also updated based on these new vehicles. Having finished the implementation of the last plan, the algorithm stops. The pseudocode of the Global Planning heuristic is given in Algorithm 7.

5.4 Enhancing Local Search

In addition to the three construction heuristics, a following heuristic is also employed to improve the solutions. The improvement heuristic is a Local Search algorithm that tries to enhance the benefit of a given platoon routing S by updating a single vehicle path. Having a platoon routing for a set of vehicles, this Local Search algorithm seeks to find the optimal path for one of the vehicles but every other vehicle routing remains fixed. In this attempt, all the feasible routes found by Algorithm 1 are examined to find the one which leads to a lower cost in case that the routes of other vehicles are frozen and do not change. Pseudocode for this Local Search algorithm can be seen in Algorithm 8. We iterate over the vehicle paths in lexicographic order and improve the path of one vehicle at a time. This Local Search terminates if no path can be improved anymore, in other words, when a local optimum is reached.

Algorithm 7: Global Planning heuristic

Data: The set of vehicles V, the road network, i.e. N and E
Result: Routing of all vehicles from the origin to destination with good platooning benefits

for $n \in N$ **do**
  - Find vehicles for which n is feasible and put them in a subset V'_n.
  if $V'_n \neq \emptyset$ **then**
    - Route the vehicles of V'_n to join at node n and find their splitting nodes based on Algorithm 5.
    - Save the routing as p_n and put it in a set called P.
  end
end
- Sort the node plans in P, firstly, based on higher saving, and secondly, based on lower sum of partial saving contributions of their vehicles in the other plans.
while $P \neq \emptyset$ **do**
  - m= the plan or node index which is the first in P.
  - Implement p_m.
  - Update vehicle of V'_m by dividing them into three parts: from all the origins into m, a unified vehicle (fixed) up to the splitting nodes and finally from the splitting nodes to the all destinations.
  - Eliminate p_m from P.
  forall $n \neq m \in P$ **do**
    - Find vehicles for which n is feasible and put them in a subset V'_n.
    - Route the vehicles of V'_n to join at node n and find their splitting nodes based on Algorithm 5.
    - Save the routing as p_n and put it in P.
  end
  - Sort the node plans in P by the two criteria.
end

5.5 Results and Comparisons

To test and compare the capabilities of the three introduced heuristics, we employ them with the complementary Local Search to solve test samples on the six networks. They are those with 10 to 100 vehicles that are solved with CPLEX in the previous chapter, and moreover, other examples including 200 and 500 vehicles. 20 instances for each size are solved with the three heuristics. The upper bounds found by the CPLEX are also good measures, which are shown to assess the performance of our heuristics. The heuristics are coded in MATLAB (MATHWORKS).

Figures 5.4 to 5.15 show the saving upper bounds (UB), savings obtained with the 3 heuristics, namely Best Pair (BP), Hub (H) and Global Planning (GP), and the corresponding solution times by box plots. The averages of savings, times and optimality gaps are presented in Tables 5.2 to 5.7. The characteristics of instances are shown like in the previous chapter: firstly, comes "S" with the number of vehicles, and then, the solution methodology, either "UB" that means the upper bound, or one of the three heuristics of this chapter shown as "BP", "H" or "GP". For example "S100 BP" means samples with 100 vehicles solved by the Best Pair heuristic.

The complementary information about the results on each network is as follows: Figure 5.4 shows the box plots of the saving percentages of upper bounds and

Algorithm 8: Pseudocode for Local Search algorithm

Data: A graph G, a platoon routing S, a vehicle and its data
Result: A platoon routing in S' with a cost lower than, or equal to the cost of S

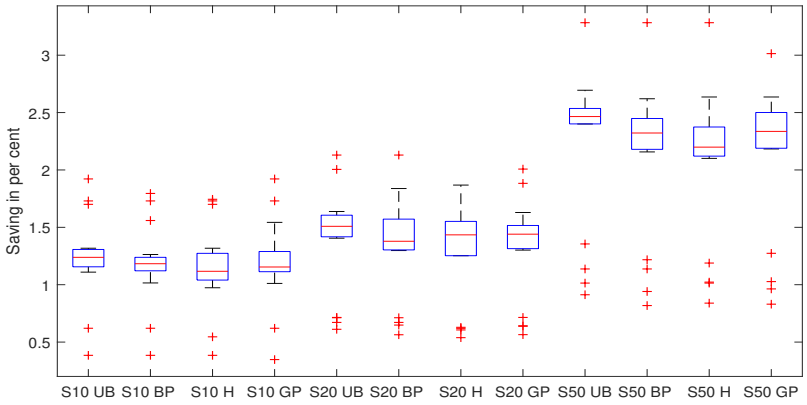
1 **while** *An improvement (lower cost) is possible by changing the route of any vehicle* **do**
2 - Lexicographically, investigate changing the route of one vehicle in the list by choosing another route from its feasible routes obtained by Algorithm 1, freeze the routes of other vehicles and calculate the benefit of each changing choice by implementing Algorithm 2.
3 - Find the changing corresponding to the maximum benefit and implement it.
4 **end**

executions of the heuristic methods for the 20 instances of each size on the Chicago network, and Figure 5.5 depicts their execution times. The means of results are summarised in Table 5.2. For S10, the average savings of BP, H and GP methods are 1.18%, 1.10%, 1.18% and their average execution times are 295, 136 and 150 seconds, respectively. Optimality gaps are 0.06%, 0.14% and 0.06%. The average saving of S20 instances are 1.34%, 1.25% and 1.34%, which are 0.07%, 0.16% and 0.07% deviated from the optimality, obtained on average in 383s, 218s and 239s. Solving the S50 example, we attained average savings of 2.14%, 1.99% and 2.15% in 513s, 361s and 377s by the BP, H and GP approach. These savings are on average 0.12%, 0.27% and 0.12% away from the optimal savings. For S100, the savings of the methods are 3.27%, 2.99% and 3.26%, which are obtained in mean times of 741s, 583s and 599s. Since we could not achieve the optimal results but only the upper bounds by the exact solution process of S100 samples in the previous chapter, it can be stated that the obtained savings by the 3 heuristics are 0.20%, 0.48% and 0.21% distant from the upper bounds. In terms of S200, the average savings of the approaches are 3.93%, 3.69% and 4.00%. The average solution times are 1092s, 937s and 951s, and since no upper bound can be attained, no information about the optimality gab or deviation from the upper bounds can be reported. Finally, in all of the S500 instances, none of the 3 algorithms can terminate within our time limit of 30 min (1800s), however, the best found solutions give mean savings of 5.32%, 4.98% and 5.39%.

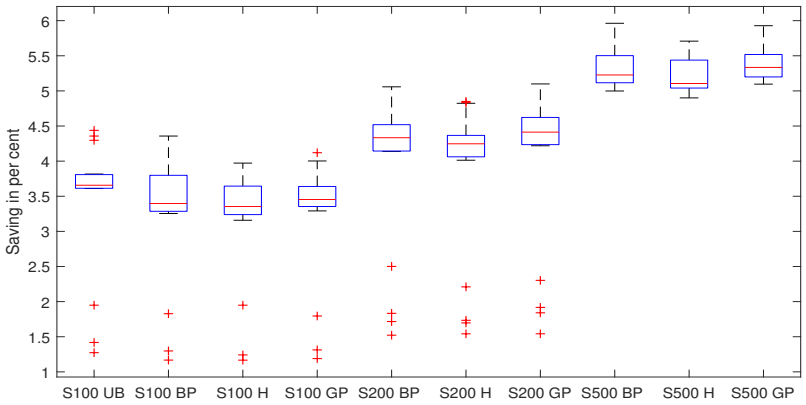
TABLE 5.2: The average savings, solution times and optimality gaps for the Chicago network with the heuristic methods

Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	BP	Hub	GP	BP	Hub	GP	BP	Hub	GP
S10	1.18	1.10	1.18	295	136	150	0.06	0.14	0.06
S20	1.34	1.25	1.34	383	218	239	0.07	0.16	0.07
S50	2.14	1.99	2.15	513	361	377	0.12	0.27	0.12
S100	3.27	2.99	3.26	741	583	599	0.20*	0.48*	0.21*
S200	3.93	3.69	4.00	1092	937	951	-	-	-
S500	5.32	4.98	5.39	1800	1800	1800	-	-	-

Figures 5.6 and 5.7 show the savings and execution times obtained by our heuristic approaches on German networks, and the averages are given in Table 5.3. The average saving percentages of BP, H and GP for S10 are 1.09%, 1.01% and 1.08%, and the mean execution times are 303s, 146s and 160s. The optimality gaps are 0.05%, 0.13% and 0.06%. For S20, our heuristics provide 1.31%, 1.22% and 1.32%



(A) Instances with 10, 20 and 50 vehicles

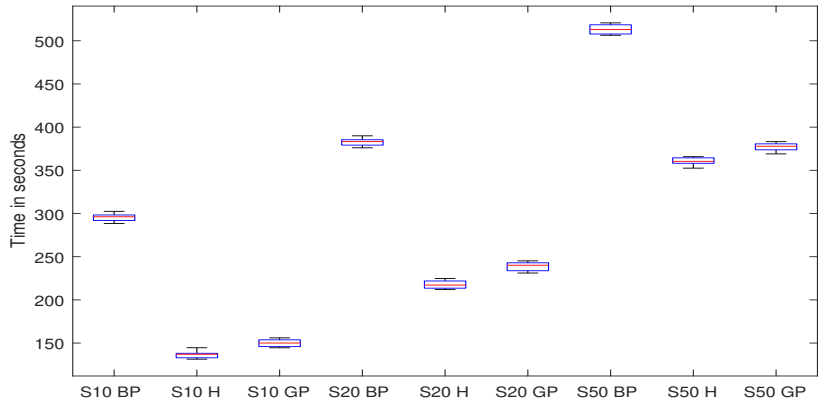


(B) Instances with 100, 200 and 500 vehicles

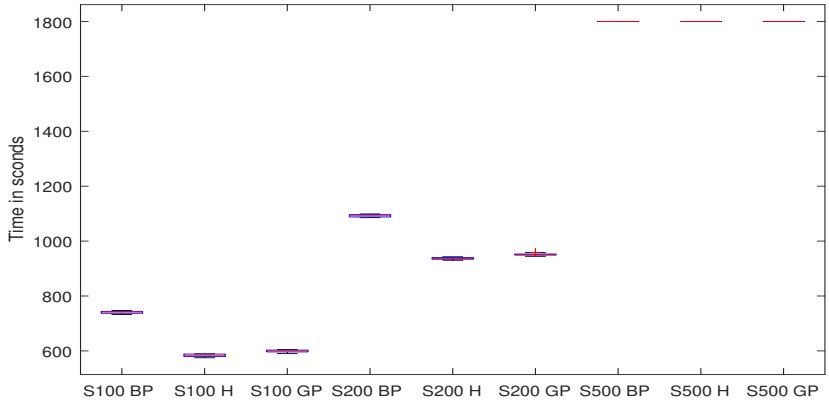
FIGURE 5.4: Upper bounds (UB) and savings with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) plus the complementary Local Search (LS) run after each on the Chicago network

saving, which are 0.09%, 0.18% and 0.08% distant from the optimal savings. These are obtained on average in 395s, 238s and 250s. S50 instances are tackled in 531s, 380s and 396s resulting in 1.99%, 1.85% and 2.01% savings by the three methods, respectively. The optimality gaps are 0.14%, 0.28% and 0.12%. Considering S100 results, 3.30%, 3.07% and 3.31% savings are obtained by elapsing 772s, 618s and 630s using the BP, H, GP algorithm. These results are 0.24%, 0.46% and 0.23% deviated from the upper bounds. From here, there is not any reference as the upper bound (UB) or optimal result, so only the savings and execution times can be reported. We have 3.78%, 3.61%, 3.86% saving in 1149s, 991s and 1005s for S200, and for S500, 5.16%, 4.77% and 5.21% saving in the situation that none of the three heuristics can stop within the time limit.

The box plots of savings and times corresponding to the Swedish network are shown in Figures 5.8 and 5.9. Likewise, Table 5.4 presents their averages. In S10 cases, the average savings are 0.90%, 0.82% and 0.91%, the average execution times are 315s, 158s and 171s, and the average optimality gaps are 0.05%, 0.14% and 0.05% for BP, H and GP, respectively. In S20 cases, the average savings of 1.05%, 0.99% and 1.06% are obtained. These results, that are 0.08%, 0.13% and 0.06% distant from the optimality, are achieved on average in 410s, 253s and 269s. For S50, the savings of the three heuristics amount to 1.85%, 1.72% and 1.85%, the optimality gaps are 0.13%, 0.26% and 0.12%, and these are obtained on average after 565s, 413s



(A) Instances with 10, 20 and 50 vehicles



(B) Instances with 100, 200 and 500 vehicles

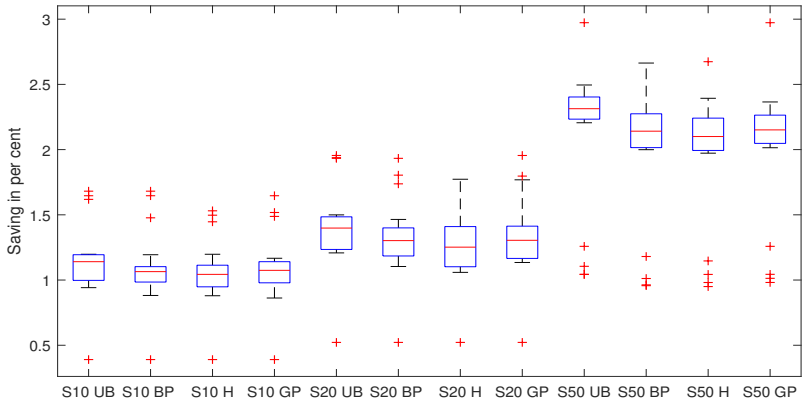
FIGURE 5.5: Execution time of solving instances on the Chicago network with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) with the complementary Local Search (LS) run after each

TABLE 5.3: The average savings, solution times and optimality gaps for the German network with the heuristic methods

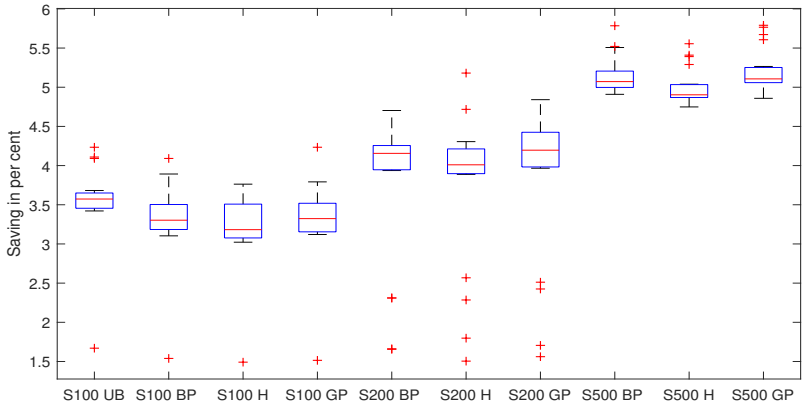
Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	BP	Hub	GP	BP	Hub	GP	BP	Hub	GP
S10	1.09	1.01	1.08	303	146	160	0.05	0.13	0.06
S20	1.31	1.22	1.32	395	238	250	0.09	0.18	0.08
S50	1.99	1.85	2.01	531	380	396	0.14	0.28	0.12
S100	3.30	3.07	3.31	772	618	630	0.24*	0.46*	0.23*
S200	3.78	3.61	3.86	149	991	1005	-	-	-
S500	5.16	4.77	5.21	1800	1800	1800	-	-	-

and 426s. Considering the results of S100 instances, we have attained 3.13%, 2.95% and 3.13% average savings in 817s, 668s and 679s, and the results are averagely 0.22%, 0.41% and 0.22% away from optimality. For S200, after elapsing the average computational times of 1218s, 1063s and 1080s, the savings corresponding to the three heuristics are 3.83%, 3.56% and 3.83%, respectively. Finally, the average of the best found solutions for S500 are 5.06%, 4.67% and 5.11%.

Figures 5.10, 5.11 and Table 5.5 are dedicated to the savings and computation



(A) Instances with 10, 20 and 50 vehicles



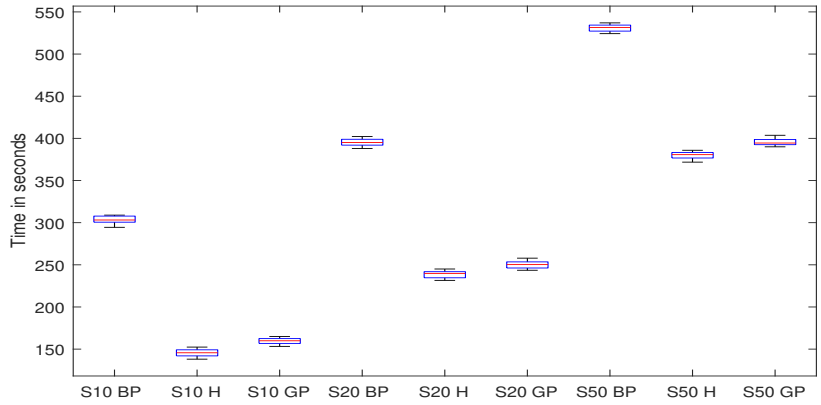
(B) Instances with 100, 200 and 500 vehicles

FIGURE 5.6: Upper bounds (UB) and savings with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) plus the complementary Local Search (LS) run after each on the German network

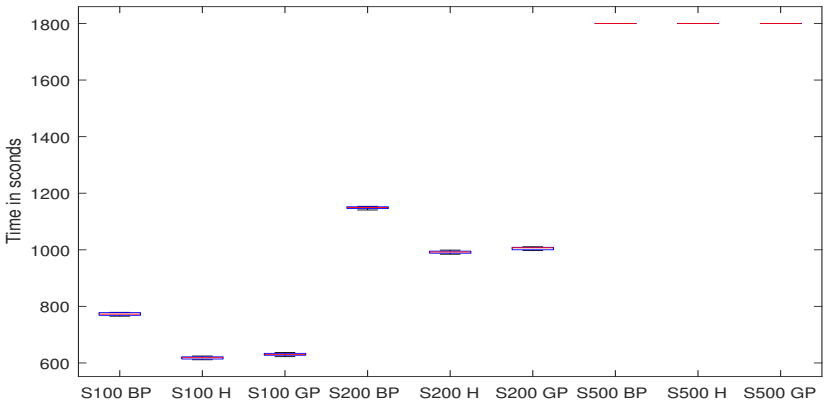
TABLE 5.4: The average savings, solution times and optimality gaps for the Swedish network with the heuristic methods

Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	BP	Hub	GP	BP	Hub	GP	BP	Hub	GP
S10	0.90	0.82	0.91	315	158	171	0.05	0.14	0.05
S20	1.05	0.99	1.06	410	253	269	0.08	0.13	0.06
S50	1.85	1.72	1.85	565	413	426	0.13	0.26	0.12
S100	3.13	2.95	3.13	817	668	679	0.22*	0.41*	0.22*
S200	3.83	3.56	3.83	1218	1063	1080	-	-	-
S500	5.06	4.67	5.11	1800	1800	1800	-	-	-

times of the examples on the Grid10 network. For S10, good average savings of 1.23%, 1.13%, 1.24% are achieved with BP, H and GP methods. The savings are corresponding to the average gaps of 0.07%, 0.17% and 0.06% from the optimality. These are obtained averagely in 321s, 161s and 173s. For S20, the average savings are 1.42%, 1.33% and 1.43%, the average optimality gaps are 0.24%, 0.33% and 0.23%, and the average execution times are 410s, 258s and 271s. For S50, the average savings of the heuristics are 2.50%, 2.32% and 2.51% obtained in average times of



(A) Instances with 10, 20 and 50 vehicles

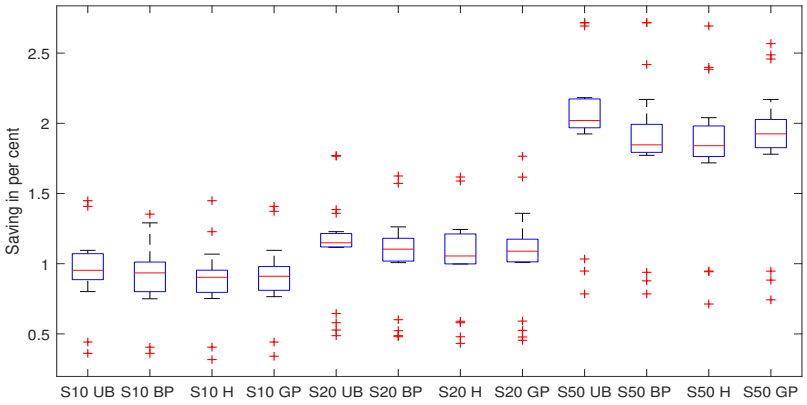


(B) Instances with 100, 200 and 500 vehicles

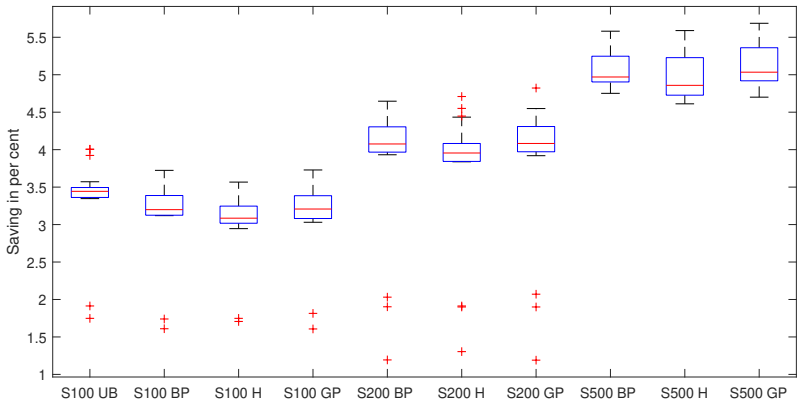
FIGURE 5.7: Execution time of solving instances on the German network with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) with the complementary Local Search (LS) run after each

563s, 416s and 430s. The savings are on average 0.18%, 0.36% and 0.17% distant from the optimal ones. The solutions of S100 samples by the three heuristics give average savings of 3.25%, 3.06% and 3.27%, which are 0.37%, 0.56% and 0.36% away from the upper bounds and reached after averagely 818s, 673s and 684s being elapsed. In case of S200, the average savings and times are 4.56%, 4.17% and 4.56%, and 1222s, 1072s and 1088s. We do not have any upper bound reference in this size. At last, the best results of the three heuristics applied to S500 samples provide 5.34%, 5.01% and 5.43% saving but the algorithms do not reach their termination.

Figures 5.12 and 5.13 include the box plots of savings and times resulted from implementing BP, H and GP algorithms on Grid30 network. The summary of results is presented in Table 5.6. S10 solutions show average savings of 1.17%, 1.09% and 1.18% corresponding to the optimality gaps of 0.07%, 0.15% and 0.06%. These results are achieved after 338s, 193s and 202s of computational time. For S20, the average savings are 1.46%, 1.31% and 1.46%, the average times are 445s, 306s and 320s, and the average optimality gaps are 0.06%, 0.20% and 0.05%. In solving S50 instances, we obtain 2.41%, 2.20% and 2.40% average saving in average computational times of 647s, 491s and 506s with 0.15%, 0.36% and 0.16% mean gap from optimality. For S100, after elapsing 954s, 793s and 807s, we locate solutions with average savings of 3.20% with BP, 3.02% with H and 3.24% with GP. The average gaps from the UBs for the three algorithms are 0.25%, 0.43% and 0.21%, respectively. In terms of S200



(A) Instances with 10, 20 and 50 vehicles



(B) Instances with 100, 200 and 500 vehicles

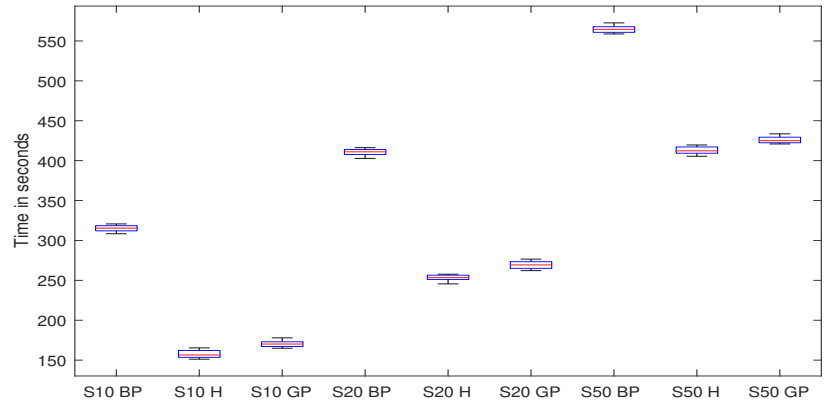
FIGURE 5.8: Upper bounds (UB) and savings with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) plus the complementary Local Search (LS) run after each on the Swedish network

TABLE 5.5: The average savings, solution times and optimality gaps for the Grid10 network with the heuristic methods

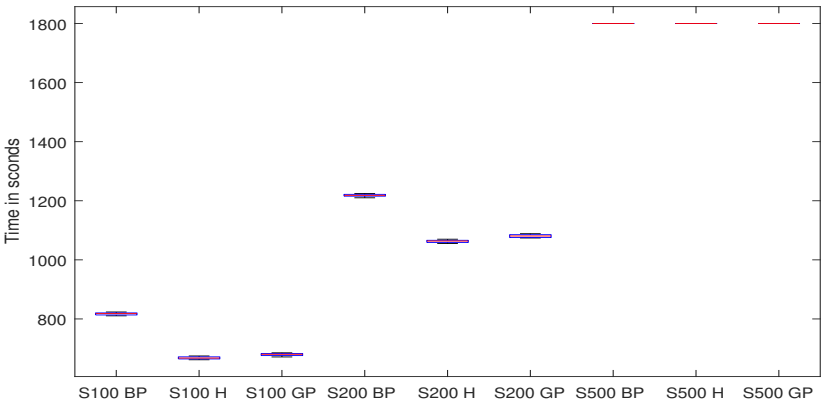
Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	BP	Hub	GP	BP	Hub	GP	BP	Hub	GP
S10	1.23	1.13	1.24	321	161	173	0.07	0.17	0.06
S20	1.42	1.33	1.43	410	258	271	0.24	0.33	0.23
S50	2.50	2.32	2.51	563	416	430	0.18	0.36	0.17
S100	3.25	3.06	3.27	818	673	684	0.37*	0.56*	0.36*
S200	4.56	4.17	4.56	1222	1072	1088	-	-	-
S500	5.34	5.01	5.43	1800	1800	1800	-	-	-

examples, the obtained averages of saving are 4.31%, 3.97% and 4.36%, and average times are 1427s, 1275s and 1287s for the three methods, respectively. For S500, the best found average savings by the three non-terminated heuristics are 5.06%, 4.67% and 5.12%.

The last network and the hardest one is Grid50. The box plots of results on this network are shown in Figures 5.14 and 5.15. The averages are given in Table 5.7. For S10, the average savings of the three methods are 1.16%, 1.05% and 1.15%, and their



(A) Instances with 10, 20 and 50 vehicles



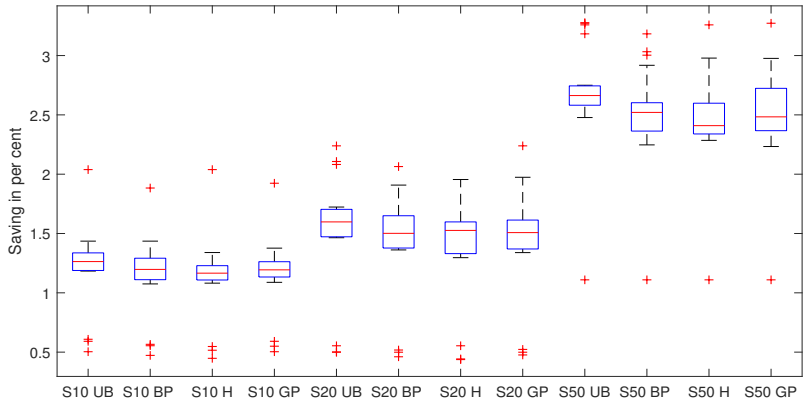
(B) Instances with 100, 200 and 500 vehicles

FIGURE 5.9: Execution time of solving instances on the Swedish network with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) with the complementary Local Search (LS) run after each

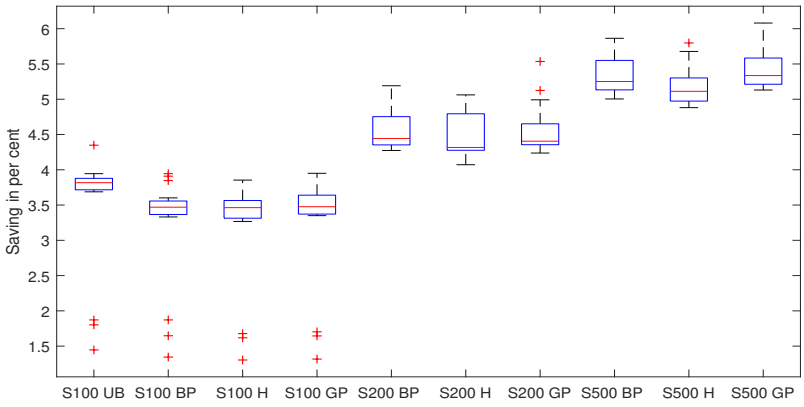
TABLE 5.6: The average savings, solution times and optimality gaps for the Grid30 network with the heuristic methods

Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	BP	Hub	GP	BP	Hub	GP	BP	Hub	GP
S10	1.17	1.09	1.18	338	193	202	0.07	0.15	0.06
S20	1.46	1.31	1.46	445	306	320	0.06	0.20	0.05
S50	2.41	2.20	2.40	647	491	506	0.15	0.36	0.16
S100	3.20	3.02	3.24	954	793	807	0.25*	0.43*	0.21*
S200	4.31	3.97	4.36	1427	1275	1287	-	-	-
S500	5.06	4.67	5.12	1800	1800	1800	-	-	-

average solution times are 373s, 221s and 238s. The corresponding optimality gaps are 0.05%, 0.16% and 0.06%. For S20 examples, the average savings of 1.24%, 1.15% and 1.25% are achieved in average times of 519s, 361s and 382s with optimality gaps equal to 0.09%, 0.18% and 0.08%, for the three methods, respectively. Solving S50 samples with our three heuristics, average savings equal to 2.32%, 2.16% and 2.35% are obtained, which are on average 0.16%, 0.32% and 0.19% distant from the optimal solutions, in 741s, 592s and 605s on average. For S100, the average savings



(A) Instances with 10, 20 and 50 vehicles



(B) Instances with 100, 200 and 500 vehicles

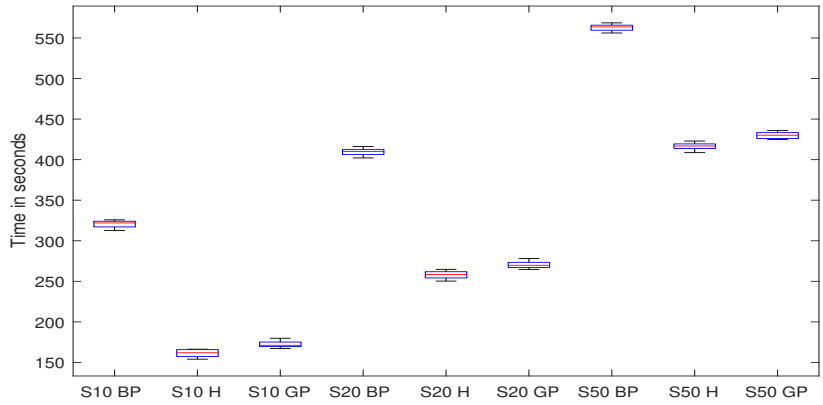
FIGURE 5.10: Upper bounds (UB) and savings with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) plus the complementary Local Search (LS) run after each on the Grid10 network

are 3.08%, 2.90% and 3.12%, the average times are 1100s, 949s and 965s, the mean gaps from UBs are 0.26%, 0.44% and 0.22%. The results of S200 samples contain average savings equal to 3.88%, 3.65% and 3.99% obtained in 1664s, 1524s and 1536s. For S500 samples, we have obtained 4.60%, 4.23% and 4.60% of average saving after exceeding the time limit of 30min.

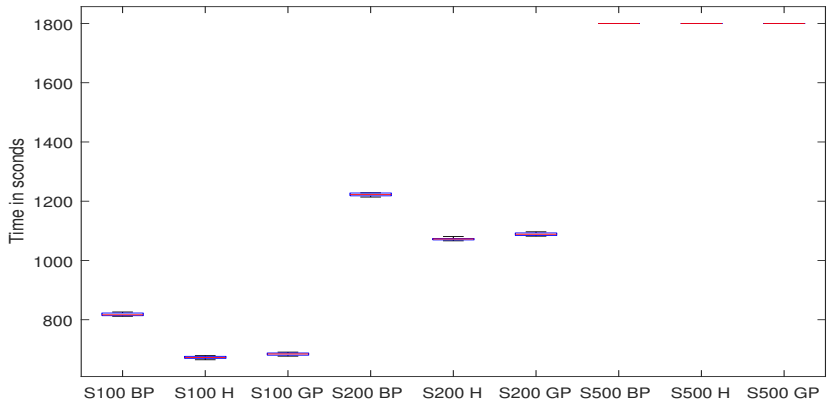
TABLE 5.7: The average savings, solution times and optimality gaps for the Grid50 network with the heuristic methods

Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	BP	Hub	GP	BP	Hub	GP	BP	Hub	GP
S10	1.16	1.05	1.15	373	221	238	0.05	0.16	0.06
S20	1.24	1.15	1.25	519	361	382	0.09	0.18	0.08
S50	2.32	2.16	2.35	741	592	605	0.16	0.32	0.19
S100	3.08	2.90	3.12	1100	949	965	0.26*	0.44*	0.22*
S200	3.88	3.65	3.99	1664	1524	1536	-	-	-
S500	4.60	4.23	4.60	1800	1800	1800	-	-	-

In general, it is observed that the totally novel approach of Global Planning (GP) can provide good results which are near to optimality for the samples with 10



(A) Instances with 10, 20 and 50 vehicles



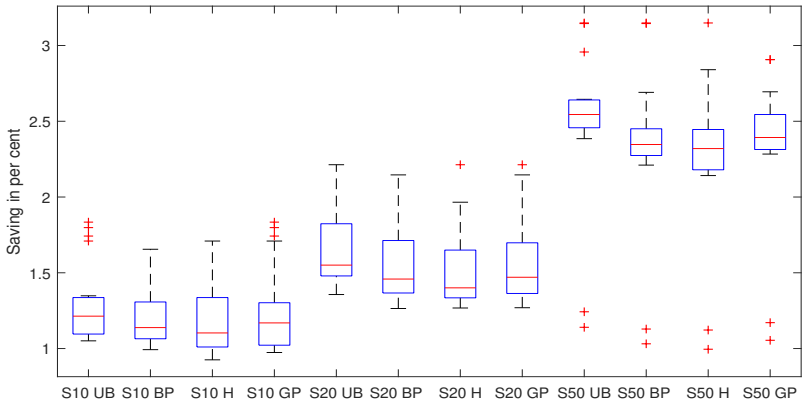
(B) Instances with 100, 200 and 500 vehicles

FIGURE 5.11: Execution time of solving instances on the Grid10 network with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) with the complementary Local Search (LS) run after each

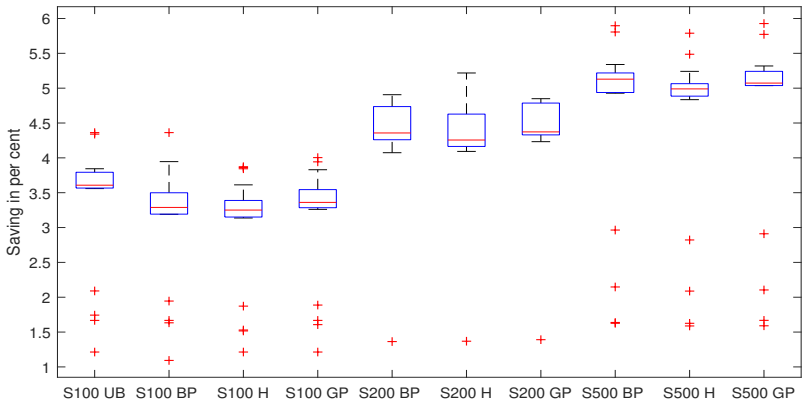
to 100 vehicles (S10 to S100), whose optimal or upper bound amounts are known. For S200, we can also achieve satisfactory savings within our time limit. However, from S500, the algorithms can not stop within the time limit. In terms of saving, in most of the cases, the GP algorithm is quite better than BP. The Hub (H) heuristic gives the weakest results among the three in terms of saving. On the other hand, regarding the execution time, H is the fastest, then is GP, and BP is the slowest algorithm. The plots of overall average savings and times of the methods regarding all the 6 road networks are shown as Figure 5.16 and 5.17.

The reason for the longer computational time of the BP is that this algorithm needs to search among all vehicle pairs and also all pairs of feasible nodes for each. However, due to a wide search, the results of this method have a good quality. On the other hand, H method breaks the whole problem into subproblems and due to smaller number of vehicles within each group and searching for only one node for splitting (joining), it is much faster than the BP. Nonetheless, the alternative GP method works with planners on the location of feasible nodes and each planner gives its platooning suggestions to the related vehicles. Therefore, this approach can have shorter run time than BP, while it can have an appropriate searching and provide solutions which are better or at least as good as those of BP.

In the section 5.7, the performance of these three algorithms are compared by using a statistical test.

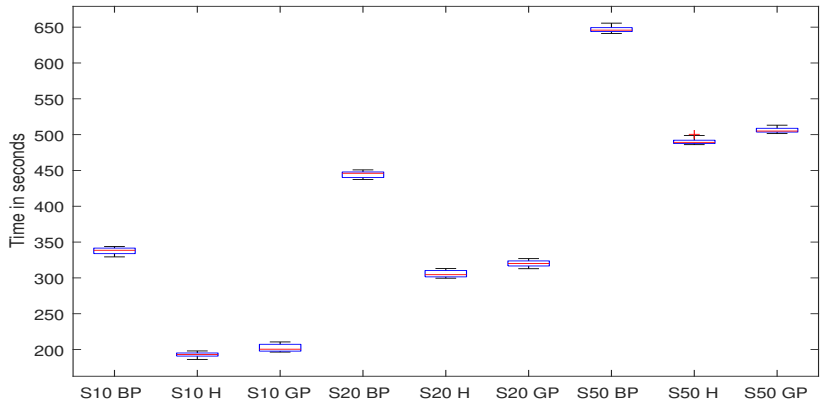


(A) Instances with 10, 20 and 50 vehicles

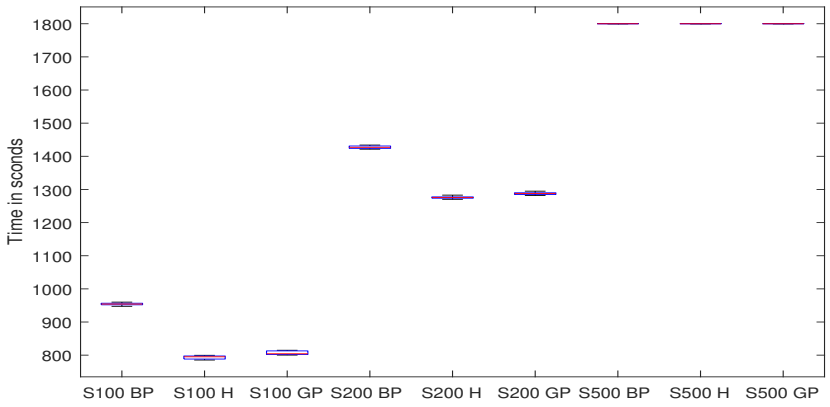


(B) Instances with 100, 200 and 500 vehicles

FIGURE 5.12: Upper bounds (UB) and savings with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) plus the complementary Local Search (LS) run after each on the Grid30 network

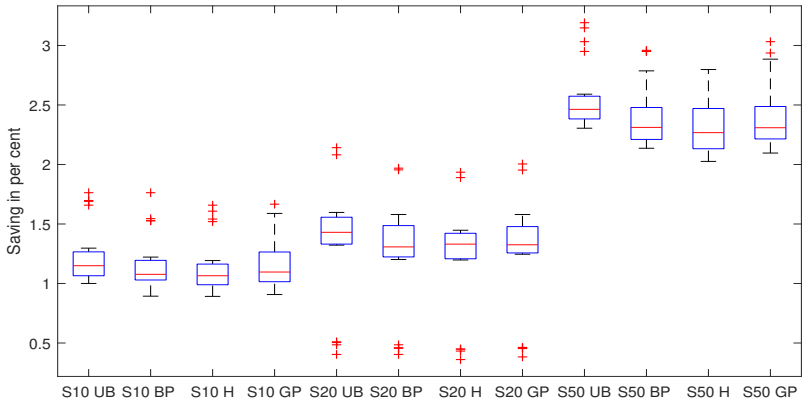


(A) Instances with 10, 20 and 50 vehicles

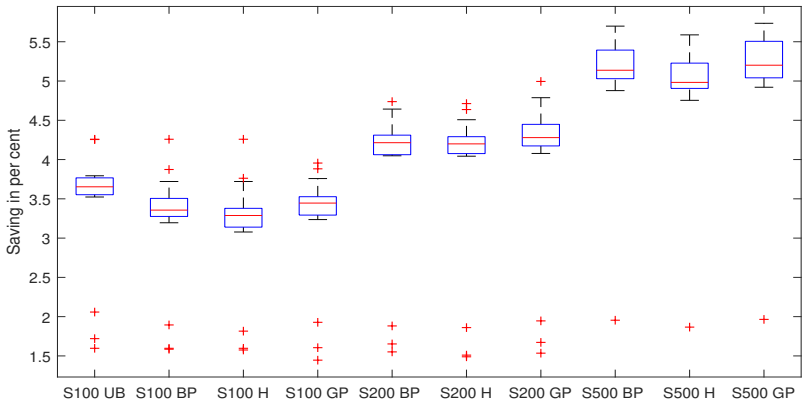


(B) Instances with 100, 200 and 500 vehicles

FIGURE 5.13: Execution time of solving instances on the Grid30 network with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) with the complementary Local Search (LS) run after each



(A) Instances with 10, 20 and 50 vehicles



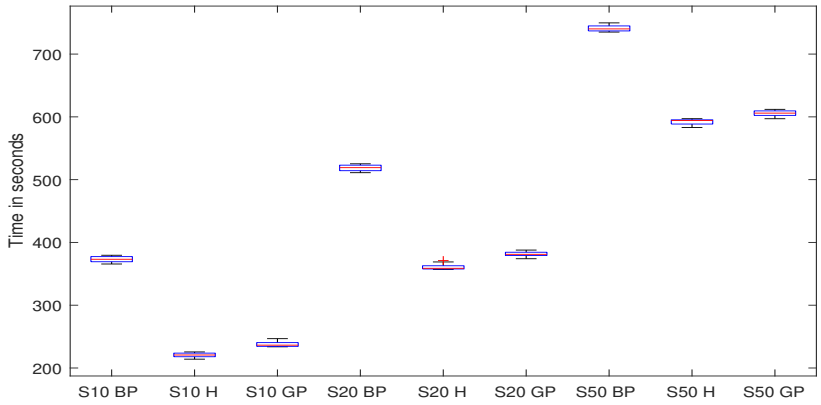
(B) Instances with 100, 200 and 500 vehicles

FIGURE 5.14: Upper bounds (UB) and savings with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) plus the complementary Local Search (LS) run after each on the Grid50 network

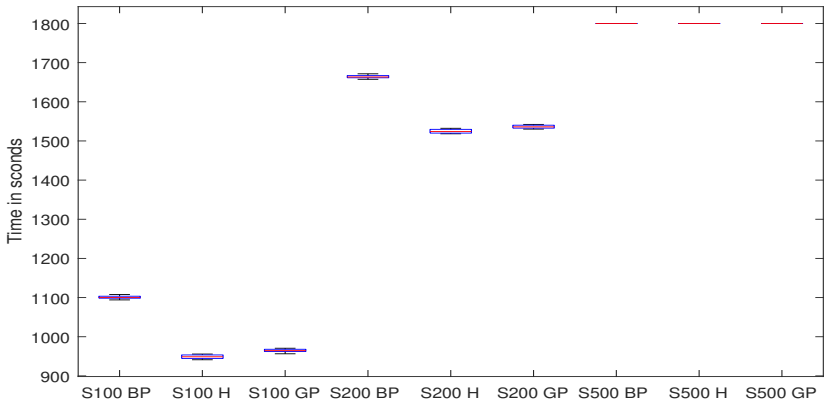
By executing the heuristics, good solutions for problem instances with up to 200 vehicles (S200) are obtained, which can not be tackled with any exact solver. However, none of the three heuristics can finish its work in S500 cases. Hence, we do not investigate any larger problem with heuristics and leave them for the next chapter to be solved by the alternative meta-heuristic solution approaches.

5.6 Effect of the Local Search

In this section, it is investigated how much of the saving rate and execution times reported in the previous section is related to the complementary Local Search (explained in Section 5.4), which is implemented at the end of the three heuristics. For this sake, each algorithm is implemented once without the Local Search, and the saving percentage and required time are measured. In this part, the results on all of the 6 networks are accumulated based on the problem size (number of vehicles). Thus, for each size, there are $20\times6=120$ results for savings and times. Since for S500 instances, none of our solution approaches stops in the first heuristic step of BP, H and GP, and they do not proceed to the Local Search (LS) step, no examination for this size is done.



(A) Instances with 10, 20 and 50 vehicles



(B) Instances with 100, 200 and 500 vehicles

FIGURE 5.15: Execution time of solving instances on the Grid50 network with the three heuristics of Best Pair (BP), Hub (H) and Global Planning (GP) with the complementary Local Search (LS) run after each

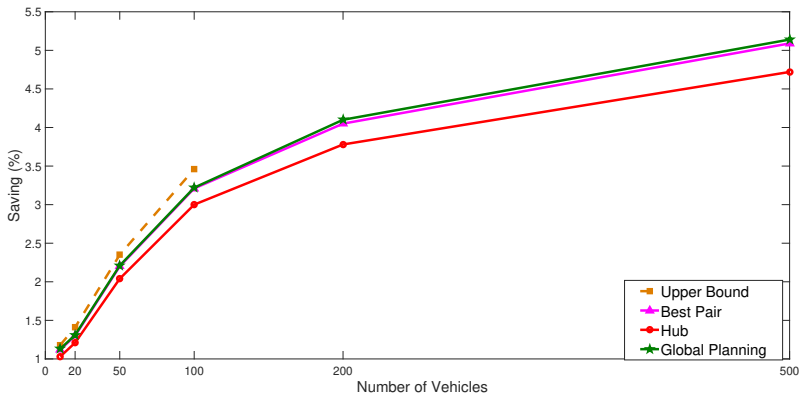


FIGURE 5.16: The overall average of the savings provided by the 3 heuristics regarding all the 6 road networks

Figure 5.18 illustrates the difference between savings of heuristics with and without the subsequent Local Search (LS) algorithm. Moreover, in Table 5.8, specific

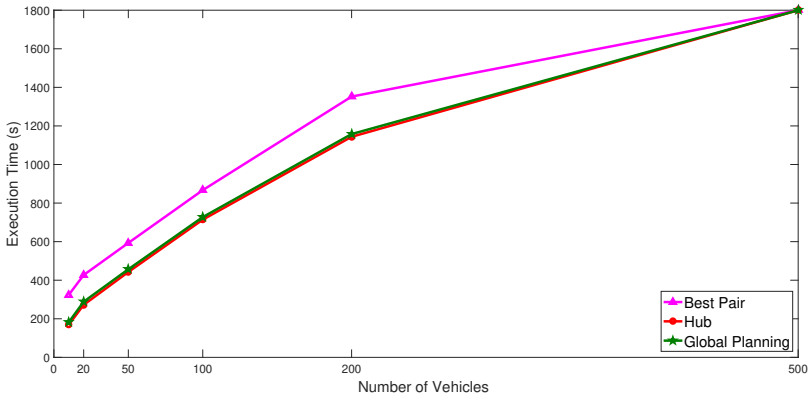


FIGURE 5.17: The overall average of the solution times of the 3 heuristics regarding all the 6 road networks

average times that belong to the Local Search part, and also its share (proportion) in the total average solution times are shown. As it is evident, by growing the problem size (Nr. of vehicles), the saving provided by LS and its run time increase. There is also a gradual rise in the share of these times. The reason is that by larger problems, there are more improvement possibilities left for the subsequent LS algorithm because the larger number of vehicles makes the problem harder to be tackled only by the first heuristics.

Heuristic / Size	Best Pair		Hub		Global Planning	
	Time(s)	Share	Time(s)	Share	Time(s)	Share
S10	49.29	0.15	30.60	0.18	26.97	0.15
S20	72.57	0.17	55.82	0.21	49.60	0.17
S50	113.91	0.19	103.43	0.23	85.40	0.19
S100	207.25	0.24	189.92	0.27	175.30	0.24
S200	339.43	0.26	322.62	0.28	296.53	0.26

TABLE 5.8: Average execution time of the Local Search executed after each heuristic and its share in the average total time

As the results show, the complementary LS algorithm can considerably improve the solutions, while it does not increase the execution time so much.

5.7 Statistical Tests

Like in the previous chapter and to have more precise comparisons between BP, H and GP heuristics, Friedman/Bergmann-Hommel tests are used to statistically compare any pair of the heuristics.

Table 5.9 contains the p-values of the pairwise tests. P-values confirm this fact that there is a significant difference between the savings of BP vs. those of H. In comparison of BP vs. GP, from S10 to S100, no significant difference in savings can be statistically proved due to the large p-values. However, it can be proved with the confidence level of 0.05 for S200 and S500. Finally, considering the very low p-values obtained for the saving comparisons between H and GP, it can be deduced that GP provides savings which are significantly better.

The p-values of execution time comparisons are given by Table 5.10. As the optimisation procedure cannot finish for S500, no p-values are shown for it. The p-values are almost zero for all comparisons of S20 to S200, which means that the

Comparison \ Size	S10	S20	S50	S100	S200	S500
BP vs. H	0	0	0	0	0	0
BP vs. GP	0.6204	0.2726	0.2724	0.3053	0.01571	0.0024
H vs. GP	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

TABLE 5.9: The p-values of the pairwise statistical comparisons of BP, H and GP savings by the Friedman test with Bergmann-Hommel post-hoc procedures

Comparison \ Size	S10	S20	S50	S100	S200
BP vs. H	6.5738e-10	0	0	0	0
BP vs. GP	1.6054e-13	0	0	0	0
H vs. GP	1.7791e-01	0	0	0	0

TABLE 5.10: The p-values of the pairwise statistical comparisons of BP, H and GP times by the Friedman test with Bergmann-Hommel post-hoc procedures

execution time of the three heuristics are significantly different. For S10, we have in comparison of BP with both H and GP very low p-values that statistically verify their time difference. Only the p-values of H time vs. GP time is quite large and do not show any significant difference in the confidence level of 0.05. Generally, the sorting of the heuristics in terms of time can be given as: H $\succ$ GP $\succ$ BP, based on the results of section 5.5 and the statistical tests of this section, noting that this sorting is from the shorter to longer run time.

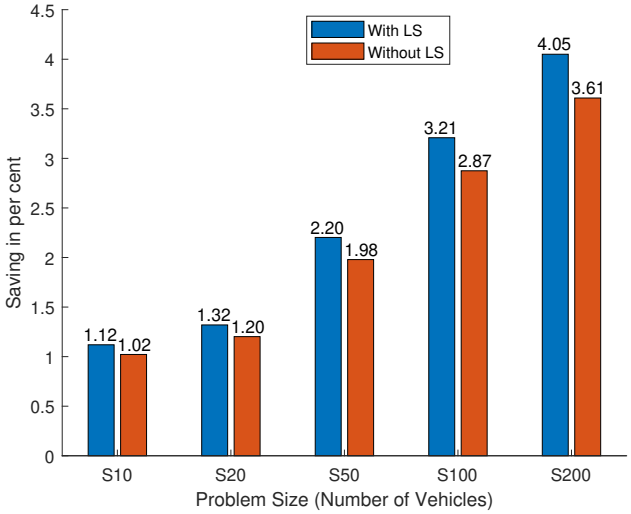
5.8 Summary

In this chapter, three heuristic solution methodologies are presented for the fuel efficient platooning (FEP) problem. Two of them, i.e Best Pair (BP) and Hub (H), are designed based on the ideas existed in the literature. However, the BP and H of this thesis are completely different and specifically designed in order to be used for an FEP problem which has time and distance constraints as well as different speed profiles. The third heuristic is Global Planning (GP) and is a totally new and efficient approach. Since this FEP is very complicated, its solution procedure is broken into two parts, namely routing and scheduling. Firstly, the routing of vehicles are constructed, and then, time scheduling, speed adjustment and objective evaluation are done with a novel approach, i.e. Algorithm 2 that works with Algorithm 3. They are important contributions in this work. These algorithms are used wherever we need to find a highly profitable scheduling for a routing and are embedded in the optimisation process of the three heuristics. For enhancement of the solutions, a Local Search (LS) method is added to end of each heuristic and it is proved that this LS is considerably useful.

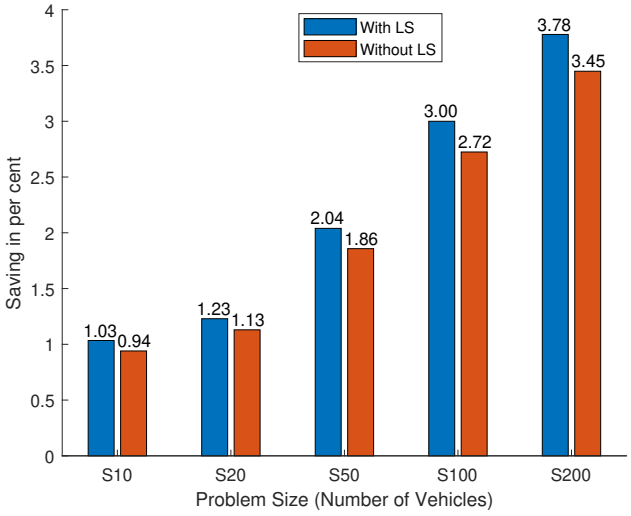
The results and their statistical tests confirm the effectiveness and efficiency of the proposed GP, since it has the advantages of both BP and H, though it do not have their disadvantages. Its solution quality is better than or quite the same as BP, however, its computational time is not considerably longer than the one of H.

Up to now, the FEP samples with up to 200 vehicles have been tackled with heuristics. Although for problems with 500 vehicles good savings are obtained, solutions with higher savings for this size are expected. This is because the heuristics cannot reach their end within our time limit and further improvements may be possible. This is investigated in the next chapter by introducing meta-heuristic

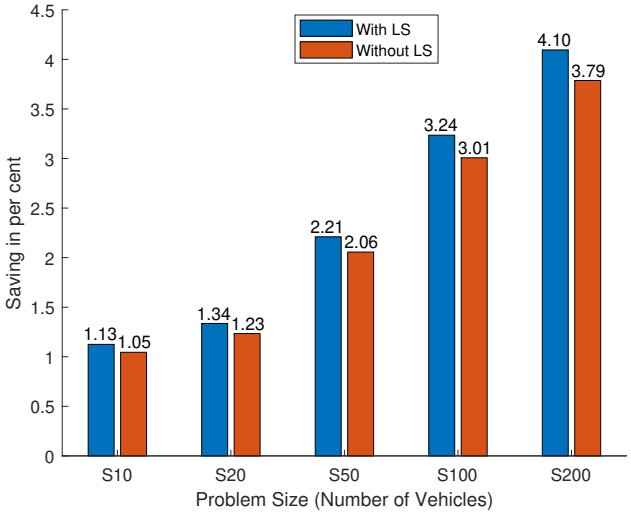
solution methodologies, which are able to deal with samples with 500 and more vehicles in relatively shorter times.



(A) Best Pair



(B) Hub



(C) Global Planning

FIGURE 5.18: Saving of the three heuristics with and without the Local Search (LS) based on the problem size

Chapter 6

Meta-Heuristic Solution Methodologies

Up to now, our Fuel Efficient Platooning (FEP) problem has been solved by an exact solver and some proposed heuristic methods. Since both of these solution types have shown some limitations and shortcomings, there is a strong need to turn to another sort of solution approaches called meta-heuristics. The name meta-heuristic is given to these methodologies because unlike the heuristic methods, that are designed for a specific problem, they can be applied to a variety of optimisation problems. These methods are inspired from natural phenomena and rules existing in the real world. Three well-known meta-heuristics, which are compatible with the features of our platooning problem, are chosen, redesigned and employed, namely: Genetic Algorithm (GA), Ant Colony Optimisation (ACO) and Particle Swarm Optimisation (PSO). The two first work with discrete solutions, while PSO has a continuous space. In this chapter: the general concept of each meta-heuristic, and then, explanations of its adaptation for the FEP problem are presented in Sections 6.1 to 6.3. The setting of the meta-heuristic parameters are explained in Section 6.4. Section 6.5 presents the results of all the three approaches, and their statistical comparisons are given in Section 6.6. Finally, Section 6.7 concludes the whole chapter.

6.1 Genetic Algorithm

6.1.1 General Concept

Genetic Algorithms (GAs) are popular optimisation methods, which use the concept of natural selection. They have been successfully applied to a wide range of operations research problems. GAs, that belong to the larger class of evolutionary algorithms (EAs), were firstly introduced by *John Holland* in the 1960s, and then, are developed by him, his students and colleagues at the University of Michigan. In this part, the general mechanism of a GA is briefly described but for details, the book of *Haupt and Haupt 2004* [37] is strongly recommended to interested readers.

A GA works on a population of candidate solutions (individuals) to an optimisation problem and tries to iteratively create better solutions by some bio-inspired operators such as selection, mutation, and crossover. In the beginning, $npop$ solutions are initialised and their objective function is evaluated. In each iteration or generation, nc solutions of the population are selected based on a selection scheme for crossover and nm solutions are randomly selected for mutation. The individuals selected for crossover exchange their parts in pairs, and so two new individuals are made. The individuals selected for mutation are randomly changed to get new solutions. The results of crossover and mutation, called offspring solutions, are evaluated and merged with the population. The solutions in the resulted pool are sorted based on their objective function value. Subsequently, the truncation is done, in which the $npop$ first (best) solutions are sent as the new generation to the next iteration and the rest is removed. This procedure continues iteration by iteration until a termination condition is met. This can be reaching a maximum number of

generations, a time limit, a satisfactory fitness level or observing no (a very low) improvement over several iterations.

6.1.2 Adaptation for the FEP problem

The general concept of GA is used to design a specific GA which is applicable to our FEP problem. First of all, a GA requires a good representation of the solution domain. This is done by encoding solutions in form of chromosomes that can be easily used throughout the optimisation process and by the genetic operators. In the proposed chromosome scheme of this work, a complete routing for vehicles are encoded by orders of their feasible nodes. We use the definition of feasible nodes for a vehicle given in the previous chapter in Subsection 5.1.1. Our chromosomes comprise rows of unequal lengths that each contain a permutation of all feasible nodes of one vehicle except its origin. Each permutation can be easily converted into a complete routing from the origin to destination of the corresponding vehicle.

This is done by putting the origin at beginning of the route as the current node, and then, finding the unvisited connected node to the current node that comes first in the permutation. If the current node is not connected to any other unvisited node, it is removed from the permutation and we jump to the next one which is connected to the current node. In each step, if the destination is reached, the procedure stops and a complete route from the origin into destination for the corresponding vehicle is constructed. This procedure is run once for each row and by this way a route is extracted for any vehicle from the chromosome. This decoding method in explained in Algorithm 9. An example of a chromosome for routing of three vehicles on a simple network and its decoding is shown in Figure 6.1.

Algorithm 9: The algorithm of decoding a chromosome into routes of vehicles

Data: The chromosome (permutations) corresponding to vehicles in V and their O^v and D^v

Result: A routing from the origin to destination for any $v \in V$

1 **for** $v = 1 : |V|$ **do**
2 - Put the elements of row v in a string called prm_v.
3 - $CN = O^v$.
4 **while** $CN \neq D^v$ **do**
5 - In prm_v , find the first unvisited node connected to CN and call it K.
6 - Remove K from prm_v.
7 **if** *There is no unvisited node connected to* K **then**
8 - Return to line 5
9 **else**
10 - $CN = K$
11 **end**
12 **end**
13 - Save the obtained order of nodes for v as its route
14 **end**

Like in the heuristic methods, firstly, the routing of vehicles are determined, and then, it is given to the Algorithm 2 of the previous chapter. So each individual or chromosome is decoded into routes of vehicles, subsequently, they are converted into a complete travel schedule, and the overall objective of FEP problem, i.e. Equation 3.21 in Chapter 3, is evaluated. This procedure is done for evaluation of all individuals in the population.

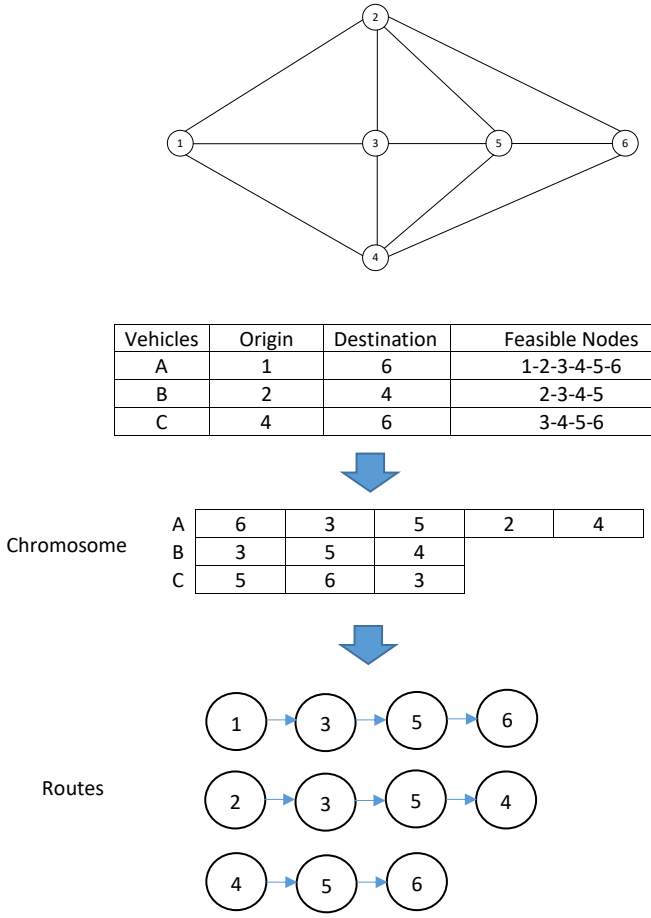


FIGURE 6.1: A Simple chromosome for routing three vehicles: A, B and C on a simple network. The road network, origin, destination and feasible nodes for vehicles, the corresponding chromosome, and its decoding into routes of vehicles are shown from the top to down.

Crossover, which is an important breeding operator, is implemented in our GA by selecting an even number, *nc*, of individuals or parents according to the tournament selection scheme (see *Blickle and Thiele, 1996* [10]). In crossover, two parents exchange their parts which are between two randomly selected points called $P1$ and $P2$. If the whole number of genes (cells containing the node labels) is n, the points are chosen by the uniform discrete distribution: *U[2,n-1]*. Figure 6.2 depicts this crossover for two simple chromosomes based on the vehicles and network given in Figure 6.1. After exchanging the parts, if any number is repeated in any row, then the chromosome must be repaired. This is because each row of chromosomes is a permutation consisting of unique elements and any number must come only once. This repairing is done by replacing the repeated elements in the part which is from the original parent (not from the other parent) with the missing numbers of the row in a random order.

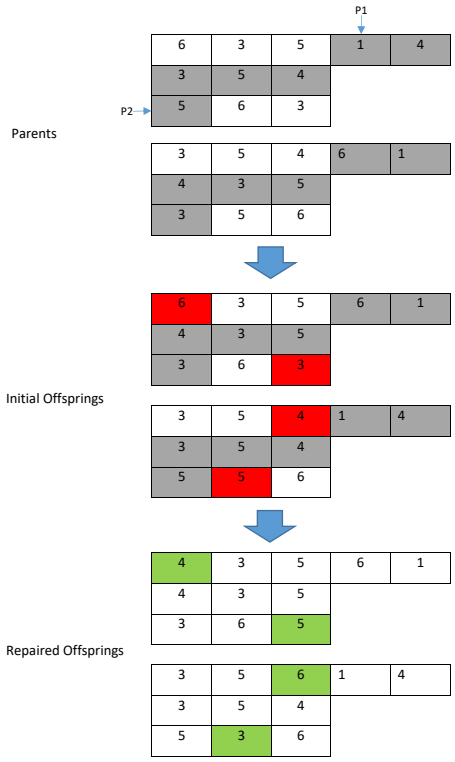


FIGURE 6.2: An example of a simple crossover on two chromosomes (parents) given for the vehicles and network of Figure 6.1. Two points $P1$ and $P2$ are randomly chosen and the parts between them shown in grey are exchanged. As in the first and last row of the initial offsprings, 6 and 3 are repeated and 4 and 5 are missing and vice versa, the repetitions which are on the original parent, shown in red, are to be replaced. This is done in the repaired offsprings and the corresponding elements are shown in green.

Mutation is another operator of a GA, that is responsible for exploration or diversification of solutions in the population. Regarding the complexity of the defined chromosome, unlike in the classical GA, we do not select any individual from the population to be mutated. Contrarily, the mutation is done by randomly creating nm new individuals.

Our GA starts with a random generation of $npop$ solutions, then in each iteration, it does crossover with $nc = 2 \times \lceil \frac{rc \times npop}{2} \rceil$ individuals chosen based on the tournament selection and creates $nm = \lceil rm \times npop \rceil$ mutations. rc and rm are the rates of crossover and mutation, receptively. It proceeds with $npop$ best

individuals out of the original population and offsprings to the next iteration. In our implementation, the GA terminates by reaching a maximum number of iterations $maxit$. The parameters of GA, i.e. $npop$, rc, rm and $maxit$ play an important role in its success, therefore, they should be appropriately set. This is the subject of Section 6.4. At the end, the GA algorithm of this section is presented as Algorithm 10. This is in terms of solution encoding and operators different from the GA presented in *Nourmohammadzadeh and Hartmann (2016)* [70].

Algorithm 10: Our proposed GA to solve FEP problem

Data: The vehicle set V, O^v, D^v, and network data N and E
Result: Fuel efficient routing, speed adjustment and time schedule for vehicles of V

- Randomly initialise $npop$ chromosomes (individuals).
- Evaluate the population by applying Algorithm 9, and then, Algorithm 2 to the resulted routing.
- It=1
while $It \leq maxit$ **do**
  - Select nc individuals and do crossover.
  - Create nm new individuals as mutations.
  - Evaluate the results of crossover and mutation called offsprings by applying Algorithms 9 and 2.
  - Pool the population and offsprings, sort them based on the fitness (objective function value).
  - Truncate the pooled population by choosing the first $npop$ solutions as the next population (generation).
  - $It = It + 1$.
end

6.2 Ant Colony Optimisation

6.2.1 General Concept

The Ant Colony Optimization (ACO) algorithm is a probabilistic searching technique for solving a wide range of optimisation problems specially those that can be stated as finding the best paths on graphs.

Belonging to the swarm intelligence family, ACO is a meta-heuristic that was initially proposed by *Marco Dorigo (1992)* in his PhD thesis [24]. It imitates the behaviour of ants seeking the best path between their colony and a source of food and was initially used to find an optimal path in a graph. However, later it was diversified to solve a wider class of numerical problems.

ACO uses this idea that in the natural world, ants of some species initially wander randomly and by finding food sources, they return to their colony while laying down pheromone trails. Next ants follow the trails by choosing their path instead of travelling at random. Upon finding food, they return and reinforce the corresponding paths. This phenomenon brings about more selection chances for the better paths. However, pheromone trails evaporate and the related routes lose their attractiveness if they are not regularly reinforced. This helps to scape from local optima.

In the application of ACO to the problem of finding the shortest path between two nodes in a graph, a population of the ants are initialised that randomly travel edge by edge from the origin to the destination. Upon reaching the destination, the length of travelled route by each ant is measured and the pheromone reinforcement of the edges of the corresponding routes are done. Shorter is a route, more

pheromone is deposited on it. This algorithm iteratively continues by the next population of ants until a termination condition is met, which is mostly defined as reaching a maximum number of iterations. This structure can be adapted to be applicable to a variety of other optimisation problems.

Nonetheless, the basic concept of ACO requires major modifications in order to solve our FEP problem. This is the subject of the next subsection.

6.2.2 Adaptation for the FEP problem

Our implementation of the ACO algorithm is somehow similar to finding shortest path on a graph but two major differences exist. Firstly, in our FEP problem, the fuel efficient paths are sought, which can be longer than the shortest path(s). Secondly, the paths must be found for a set of vehicles with different origins and destinations, and not only between one common start and end node.

To resolve these two issues, we define the ants which each represent all the vehicles. In our ACO, there are a set of $asize$ ants initially located at the origin of the first vehicle in V, v_1 (vehicles can be arranged in any order) and it is named the current vehicle CV. Each ant probabilistically traverses edge by edge and upon reaching the destination of CV, it jumps to the origin of the next vehicle and continues from there. It is now the new CV. $|V|$ different pheromone types are defined that each is corresponding to one vehicle. An ant is only influenced by the pheromone of its current vehicle or CV. Upon arrival of ant m at the destination of the last vehicle v_l, a complete solution called sol_m is constructed, which contains the routes of all vehicles. Subsequently, its objective function, $F(sol_m)$, is calculated by the aid of Algorithm 2 of the previous chapter. After each iteration, the pheromone deposits on each edge of the graph are calculated based on the below formula:

$$\Delta\tau_{ij}^{vm}(It) = \begin{cases} \frac{Q}{F(sol_m)} & if\ in\ sol_m\ vehicle\ v\ passes\ edge\ e_{ij} \\ 0 & otherwise \end{cases} \tag{6.1}$$

where the pheromone v amount which ant m deposits on edge e_{ij} in iteration It , i.e. $\Delta\tau_{ij}^{vm}(It)$, is calculated based on the objective function value of the solution provided by ant m, sol_m, in iteration It, $F(sol_m)$, and a constant value, Q, which is assumed to be equal to 1.

The pheromone update rule in each iteration based on the new deposits of ants and the last trails is the same as in classical ACO but a major difference is that the pheromone amount is calculated separately for each pheromone type v:

$$\tau_{ij}^{v}(It) = (1-\rho) \times \tau_{ij}^{v}(It-1) + \sum_{m=1}^{asize} \Delta\tau_{ij}^{vm}(It) \tag{6.2}$$

where $0 \leq \rho \leq 1$ is the evaporation rate which is applied on the last pheromone trail, $\tau_{ij}^{v}(It-1)$. The second term on the right hand side sums up the pheromone deposits of all ants in iteration It.

With regard to the pheromone related to each vehicle v on each candidate edge e_{ij}, the ant whose current vehicle is v chooses the edge in iteration It with the probability:

$$p_{ij}^{v}(It) = \frac{(\tau_{ij}^{v}(It))^{\alpha}(\sigma_{ij})^{\beta}}{\sum_{k \in a^{v}(i)} (\tau_{ik}^{v}(It))^{\alpha}(\sigma_{ik})^{\beta}} \tag{6.3}$$

where p_{ij}^{v} is the probability that v chooses edge e_{ij}. σ_{ij} is called heuristic information or desirability of edge e_{ij} and should be calculated based on the edge length. So here σ_{ij} is assumed to be $1/l_{ij}$. $a^{v}(i)$ is the set of feasible choices that truck v has if it is now in node i. $\alpha, \beta \geq 0$ are parameters that determine the relative influence of pheromone trail and edge desirability in the probability calculation.

Our ACO searches for better solutions iteration by iteration until the maximum number of iterations $maxit$ is reached. All $\tau_{ij}^{v} = 1$ is initialised at the beginning.

Algorithm 11 describes the whole procedure of this ACO. This ACO considerably differs from the solution algorithm presented in *Nourmohammadzadeh and Hartmann (2018a)* [68] mainly because that algorithm works on a single solution in each iteration.

Algorithm 11: Our proposed ACO to solve FEP problem

Data: The vehicle set V, O^v, D^v, and network data N and E
Result: Fuel efficient routing, speed adjustment and time schedule for vehicles of $V = \{v_1, v_2, ..., v_l\}$

- Initialise $\tau_{ij}^v = 1$ for all $v \in V$ and $e_{ij} \in E$
- It=1
while $It \leq maxit$ **do**
  for $m = 1 : asize$ **do**
    - Put ant_m in O^{v_1}, $CV = v_1$
    while ant_m *has not reached* D^{v_l} **do**
      - Determine the next node of ant_m from the feasible nodes of CV connected to the current node probabilistically according to the formula 6.3.
      if ant_m *has reached* D^{CV} **then**
        - Put ant_m in the origin of the next vehicle
        - $CV = next\ vehicle$ in V
      end
    end
    - Evaluate the objective of the routing constructed by ant_m, $F(ant_m)$, by Algorithm 2
    - Update pheromone trail on the edges of graph, i.e. all $e_{ij} \in E$, by using equations 6.1 and 6.2
  end
  -$It = It + 1$
end

The main parameter of this ACO are $asize$, α, β, ρ and $maxit$, which are set in Section 6.4

6.3 Particle Swarm Optimisation

6.3.1 General Concept

Particle Swarm Optimisation (PSO) is a computational method that imitates the movement of bird or fish flocks to optimise a problem. This method employs a population of agents within the solution space called particles and they are moved in each iteration according to their position and velocity. In the end, it is expected that most of the particles converge to a solution with an excellent quality.

PSO was originally introduced by *Kennedy, Eberhart and Shi* in [47] and [82]. This is another algorithm which uses the swarm intelligence concepts like ACO. For details about PSO and swarm intelligence specially the philosophical aspects, the book of *Kennedy and Eberhart* [48] is recommended.

In PSO, an initial population of particles are generated in the solution space and an initial speed is assigned to each . In each iteration, each particle is moved toward the best position that it has experienced and the global best position of all particles so far, and the algorithm continues until the termination condition is achieved. The movement is according to the particle's new speed which is calculated based on

the two mentioned factors, i.e. the best individual position and the best global position. Thus during each iteration for any particle, firstly, its speed, and secondly, its position is updated based on the new speed. The update formulas for speed and position are as below:

$$v_i(It) = v_i(It-1) + c_1\varphi_1.(p_i^{Best} - x_i(It-1)) + c_2\varphi_2(p_g^{Best} - x_i(It-1)) \quad (6.4)$$

$$x_i(It) = x_i(It-1) + v_i(It) \quad (6.5)$$

Where $x_i(It)$ and $v_i(It)$ are the position and speed vector of particle i in iteration It, respectively. These vectors for the previous iteration are $x_i(It-1)$ and $v_i(It-1)$. p_i^{Best} is the best position that particle i has experienced so far. p_g^{Best} is the best position of all particles found up to now. c_1, c_2 are learning factors and usually $c_1 = c_2 = 2$. φ_1 and φ_2 are random numbers in the range [0,1].

6.3.2 Adaptation for the FEP Problem

A big challenge in applying PSO to our FEP problem is that PSO works with a continuous domain, however, our solutions are defined as a string of feasible node labels, which are discrete. To resolve this issue, we let PSO evolve continuous solutions but convert them to an equivalent discrete form, and then, they can be evaluated like the chromosomes of GA. Firstly, the particles of PSO are defined with a structure similar to the chromosomes in GA but with this difference that the cells can be filled with any continuous value within the interval [0,1]. In the next step, the values in each row are sorted from the lowest to highest and their rank in this sorted order are obtained. According to the ranks, the corresponding node labels are entered into the cells. The rule says that the cell containing the lowest continuous amount is filled by the lowest node label, the second lowest with the second label and so on.

By this technique , a continuous PSO particle is converted to a discrete solution which can be easily decoded to routings of vehicles. Consequently, it is evaluated by Algorithm 2, which is widely used in this thesis. An important advantage over the GA chromosomes is that any particle with such a structure containing continuous values can be simply decoded to vehicles' routes without requiring any repair after the PSO movements. In Figure 6.3, this discretisation approach is shown for a continuous solution (particle) given for the same problem encoded for GA in Figure 6.1.

The PSO algorithm starts like the previous meta-heuristics by generating $psize$ random particles or solutions. Their initial velocity $v_i(0)$ for $i = 1, ..., psize$ is a random solution structure with cells containing values in the interval [0,1]. In other word, the initial velocities are in form of another random solution. In each iteration, the particles are evaluated and the global best until that iteration and the best experience of each particle are updated. The particles are moved towards their best personal and the global best solution found until that iteration, using formulas 6.4 and 6.5. If by the movement any element of any particle goes beyond the defined interval [0,1], it is reflected into the allowable space by returning the rest of its movement in the opposite direction. Like the previous two algorithms, after reaching a maximum number of iterations $maxit$, the algorithm stops. This PSO is introduced in Pseudocodes in Algorithm 12. The PSO presented in *Nourmohammadzadeh and Hartmann 2018b* [69] is another variant because it treats the constraints by adding penalties to the objective function.

The performance of this PSO is also strongly dependent on the values of its main parameters, i.e. $psize$ and $maxit$, which are tuned in the next section.

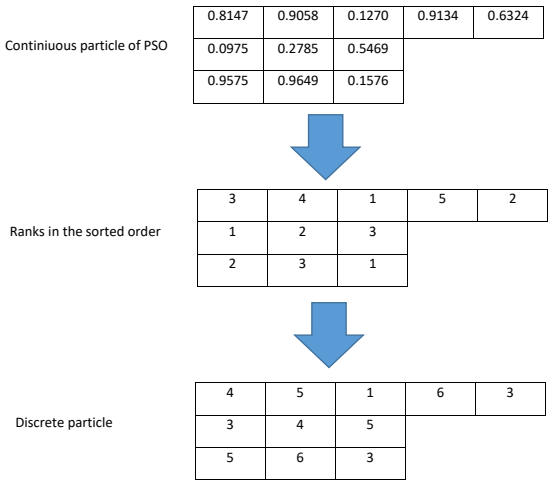


FIGURE 6.3: An example for converting a continuous particle, defined for the problem of Figure 6.1, into an FEP problem solution. On the top, the PSO particle is shown, then the ranks of values in the sorted order are entered. Finally, in the bottom, the acceptable discrete solution, in which the ranks are converted to the corresponding feasible node labels, is introduced. For example, the first element in the first row of the PSO particle is the third in the ranked order of the continuous values of this row. This third rank is corresponding to the label 4 (1 is the first, 3 the second and then 4).

Algorithm 12: Our proposed PSO to solve the FEP problem

Data: The vehicle set V, O^v, D^v, and network data N and E
Result: Fuel efficient routing, speed adjustment and time schedule for each vehicle of V

- Initialise $psize$ random particles and assign them a random velocity
- It=1
while $It \leq maxit$ **do**
  for $i = 1 : psize$ **do**
    - Evaluate particle i by firstly discretisation, and subsequently, applying Algorithm 2.
    - Update the best experience of particle i (p_i^{Best}) and the global best result (p_g^{Best}).
    - Update the velocity of particle *i* by formula 6.4
    - Update the position of particle *i* by formula 6.5
  end
  - $It = It + 1$
end

6.4 Parameter Setting

The parameters of a meta-heuristic play a crucial role in its ability to solve a problem. Hence, it is worthwhile to investigate into parameter setting of any meta-heuristic. However, finding the best or optimal combination of parameter values is itself a very complicated problem and sometimes even harder than our original optimisation problem, which the meta-heuristic seeks to solve. There have been some attempts in finding or designing efficient approaches for parameter tuning. One is using Design of Experiments (DOE) methodologies like Taghuchi [76] and Response Surface Method (RSM) [13]. Another approach, called meta-optimsation, is to employ another metha-heuristic to optimise the parameter values of the main meta-heuristic. Although meta-optimisation methods can extensively search in the space of parameter values, they are very time consuming because each solution of the first meta-heuristic, which is responsible for parameter setting, requires a run of the second (main) metaheuristic, which optimises the original problem. This issue shows itself specially in the case of our FEP problem because each solution needs running a couple of algorithms to be evaluated requiring a considerable amount of time. However, DOE methods can provide good results in a much shorter time. We choose the Reponse Surface Method (RSM) for parameter setting of our three meta-heuristics because it searches within a whole interval determined for each parameter, unlike the Taghuchi method, which examines the combination of some limited values (levels) of parameters.

As mentioned, there are 4 parameters in GA, namely pop, nc, nm and $maxit$, 5 parameters in ACO, namely $asize$, α, β, ρ and $maxit$, and 2 parameters in PSO, namely $psize$ and $maxit$, to be set at their best value that can be found. For each parameter, a searching interval is considered in which the best values are most probable to be found based on the empirical experience.

The parameter setting is done separately for each problem size, i.e number of vehicles from 10 to 2000. Based on a full RSM design, 31 experiments are needed for GA having four factors, while 52 experiments have to be run for ACO because of the five factors and 13 experiments for PSO due to only 2 factors. For each combination of parameter values of RSM, we run 5 different instances and calculate their average as the response factor. Since among the 6 networks the hardest one or worst case is Grid50, all of the 5 instances are implemented on this network. This is because doing all numerous RSM experiments on all the networks is very time consuming and we hope that the tuning for the hardest network is successfully usable for the easier ones. All meta-heuristic solution approaches of this chapter are implemented in MATLAB (MATHWORKS) on the same computers used for exact and heuristic solution experiments of the previous chapters.

The RSM is done in Minitab17 software [84]. In Tables 6.1, 6.2 and 6.3, the considered intervals (given in brackets under the the parameters' names) and the finally chosen values of parameters are shown for the GA, ACO and PSO, respectively.

As an example, the RSM implementation for setting the PSO parameters to solve S1000 instances is presented in this part. Based on the full design for two parameters (factors), their values are mapped (coded) in the interval [-1,4142, 1,4142]. Since $psize$ and $maxit$ are discrete, whenever it is required in decoding, they are rounded to the next integer values. In Table 6.4, the 13 RSM experiments are shown with the coded and real values of parameters according to their original intervals, i.e. $psize$ in [50,250] and $maxit$ in [50,550], and the responses (average savings).

Figure 6.4 illustrates the 3D surface plot of the relation between the two PSO parameters and the average saving (response factor) of this RSM implementation. In the bottom, the optimised values of parameters are shown. According to the RSM optimisation, the coded values of 1.2999 and 0.8211 are chosen for $psize$ and $maxit$, which are equivalent to the real values of 193 particles and 396 iterations.

In the end, it is worth-nothing that another factor in running of each experiment is the time limit of 30min. It restricts the execution time of any solution approach

TABLE 6.1: Parameter tuning of the GA by response surface method

Size \ Parameter	$npop$ [50,250]	nc [0.4,0.8]	nm [0.2,0.8]	$maxit$ [50,550]
S10	75	0.48	0.28	78
S20	95	0.52	0.33	126
S50	112	0.55	0.34	156
S100	136	0.58	0.36	188
S200	148	0.62	0.37	207
S500	175	0.63	0.38	280
S1000	202	0.65	0.40	392
S2000	265	0.71	0.52	534

TABLE 6.2: Parameter tuning of the ACO by response surface method

Size \ Parameter	$asize$ [50,250]	α [0,2]	β [1,2]	ρ [0,1]	$maxit$ [50,550]
S10	92	1.31	1.20	0.09	85
S20	105	1.31	1.22	0.10	131
S50	118	1.32	1.21	0.10	160
S100	142	1.34	1.22	0.11	194
S200	155	1.35	1.23	0.10	215
S500	183	1.35	1.22	0.12	284
S1000	210	1.38	1.21	0.12	403
S2000	276	1.42	1.20	0.11	536

TABLE 6.3: Parameter tuning of the PSO by response surface method

Size \ Parameter	$psize$ [50,250]	$maxit$ [50,550]
S10	84	85
S20	101	122
S50	117	157
S100	138	188
S200	149	208
S500	178	265
S1000	193	396
S2000	272	520

in this thesis. Hence, whenever the algorithm cannot reach the final determined iteration of an RSM experiment within this time limit, the result of the last achievable iteration is reported.

6.5 Results and Comparisons

The three meta-heuristics introduced in this chapter are applied to FEP instances after being parametrically tuned. The first part of instances are exactly those solved with exact and heuristic methods in the two previous chapters. In addition, larger

TABLE 6.4: RSM Experiments for PSO

Exp. Nr.	$psize$		$maxit$		Saving % (Response)
	Real	Coded	Real	Coded	
1	100	0	250	0	5.93
2	100	0	250	0	6.05
3	171	1	428	1	6.35
4	171	1	73	-1	5.25
5	100	0	250	0	5.95
6	100	0	500	1.4142	6.20
7	0	-1.4142	250	0	5.36
8	200	1.4142	250	0	5.91
9	100	0	0	-1.4142	5.46
10	100	0	250	0	5.86
11	30	-1	73	-1	5.02
12	30	-1	428	1	5.72
13	100	0	250	0	6.05

instances which are intractable even by heuristic approaches are given to the meta-heuristics. In total, the number of vehicles in the instances ranges from 10 to 2000, and they are solved on the six networks of 3.4 of Chapter 3.

Similar to the previous experiments of this thesis, 20 different samples for any problem size are solved by the proposed Genetic Algorithm (GA), Ant Colony Optimisation (ACO) and Particle Swarm Optimisation (PSO) implemented in MATLAB. The box plots of the obtained savings and required execution times as well as the tables of averages are presented in this section. Like in the presentation of previous results, in the horizontal axis of the plots, firstly, the number of vehicles comes after "S", then the solution method is written, which is UB, i.e. upper bounds obtained by the exact solver (CPLEX) in Chapter 4, GA, ACO or PSO. For example, "S100 GA" means the (20) instances with 100 vehicles solved by the Genetic Algorithm.

Supplementary information including the average savings, average optimality (from the upper bound) gaps and solution times are given for each problem size on each network as below:

Figure 6.5 shows the savings of upper bounds and meta-heuristics for the 20 instances of each size on the Chicago network by box plots. Similarly, Figure 6.6 depicts their execution times. The averages of savings, times and optimality gaps are given in Table 6.5. For S10, the average savings of GA, ACO and PSO are 1.21 %, 1.21% and 1.20%. The Optimality gaps corresponding to these solutions are 0.03%, 0.04% and 0.03%. The average solution times of the methods are 46s, 43s and 48s, respectively. The average saving of S20 instances are 1.37%, 1.39% and 1.37%, which are on average 0.04%, 0.02% and 0.04% deviated from the optimality, obtained on average in 71s, 67s and 69s. Solving the S50 examples, we attained 2.19%, 2.22% and 2.19% average saving in 116s, 103s and 108s by the GA, ACO and PSO algorithm. These savings are on average 0.07%, 0.04% and 0.07% away from the optimal savings. For S100, the savings of the meta-heuristics are 3.34%, 3.39% and 3.30%, which are obtained in 183s, 165s and 177s on average. The optimal results are not obtained for this size but it can be stated that the obtained average saving by the 3 meta-heuristics are 0.19%, 0.14% and 0.23% distant from the upper bounds. In terms of S200, the average savings of the GA, ACO and PSO are 4.07%, 4.11% and 4.05%, and the average solution times are 306s, 276s and 294s. Since we do not even have any upper bound, no optimality or gap from the upper bound can be reported. In the S500 instances, the average savings are 5.57%, 5.66% and 5.49%, earned in 515s, 461s and 494s on average by GA, ACO and PSO, respectively. Regarding S1000 instances, GA, ACO and PSO return 6.83%, 6.97% and 6.72% average saving in the mean times of 869s, 835s and 783s, respectively. For S2000,

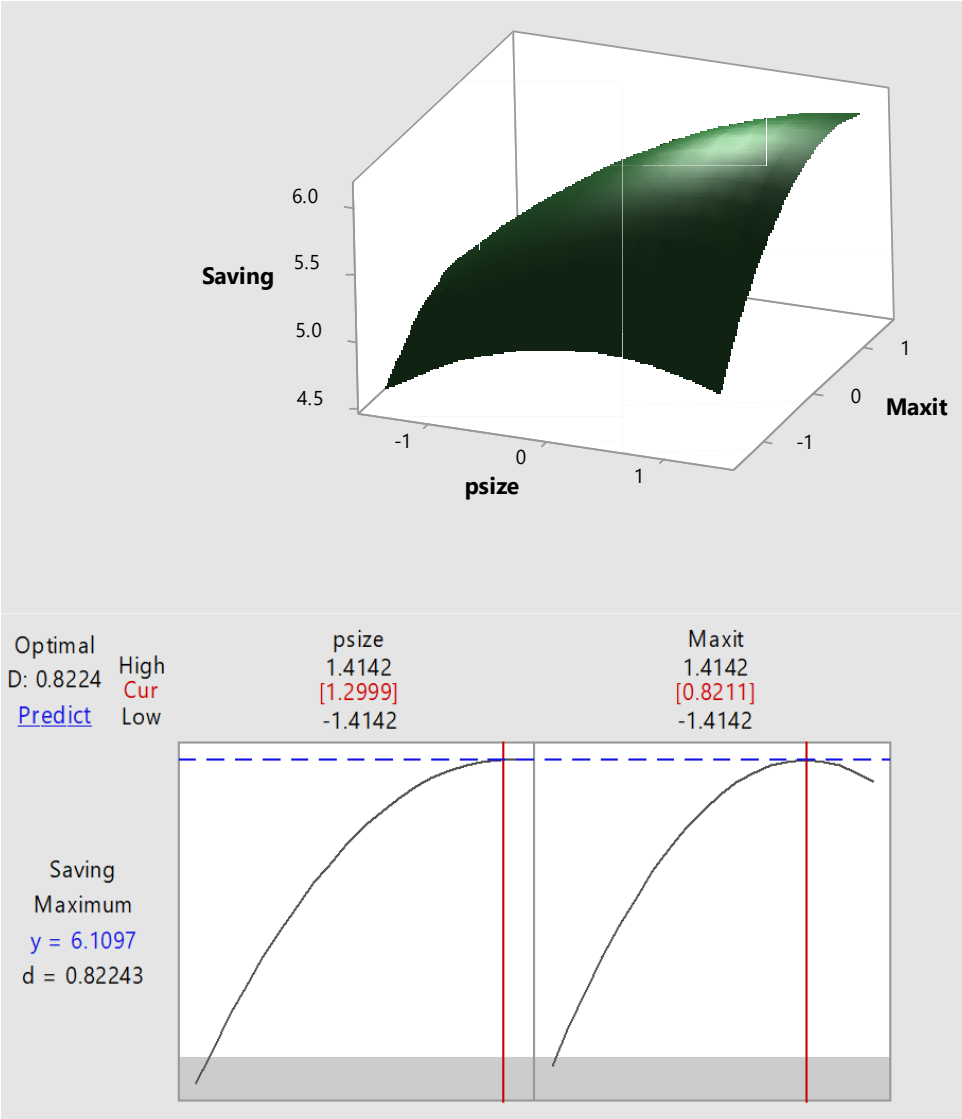
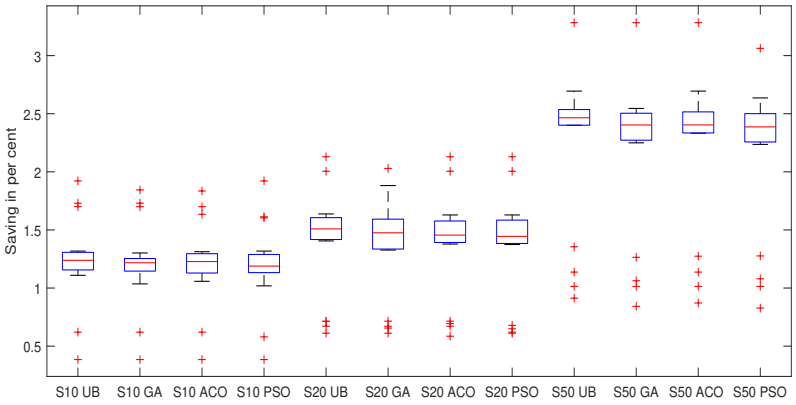


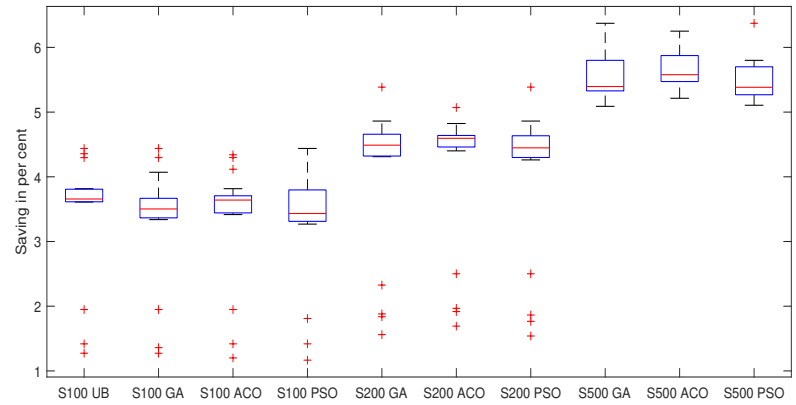
FIGURE 6.4: Surface plot of saving vs. $psize$ and $maxit$, and in the bottom, the plots of optimising the parameters' values by RSM method

which is the largest size that the meta-heuristics can reach their final iteration in dealing with it within the time limit of 30 min, GA gives 7.67% saving in 1473s, ACO gives 7.81% saving in 1328s and PSO gives 7.57% saving in 1415s on average.

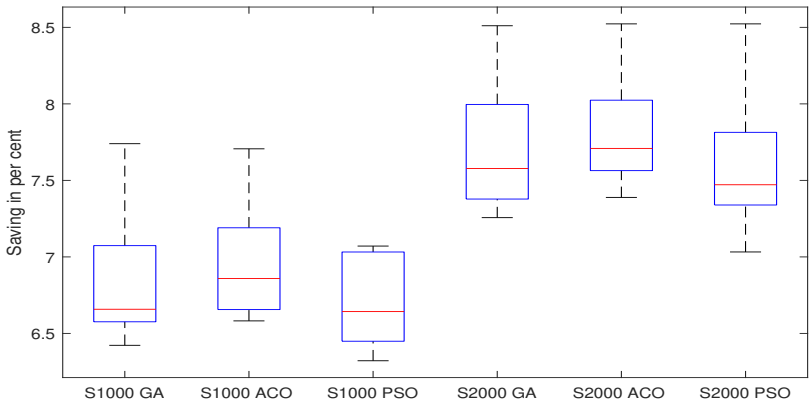
Figures 6.7 and 6.8 show the saving and execution times of meta-heuristics on the German network. Table 6.6 gives the averages. The average saving of GA, ACO and PSO for S10 are 1.12%, 1.12% and 1.10%, and the average solutions times are 54s, 49s and 49s. The optimality gaps are 0.02%, 0.02% and 0.04%. For S20, the methods provide 1.36%, 1.38% and 1.35% average saving, which are 0.04%, 0.02% and 0.05% distant from the optimal savings. These are obtained on average in 79s, 74s and 76s. S50 instances are tackled by the three methods resulting in 1.99%, 2.01% and 1.85% average savings, respectively. The execution times are 125s, 116s and 123s. The average optimality gaps are 0.14%, 0.12% and 0.28%. Considering S100 results, we obtained 3.42%, 3.46% and 3.39% average saving by elapsing 204s, 185s and 195s using GA, ACO, PSO algorithm. These results are 0.05%, 0.01% and 0.08% away from the upper bounds. For S200, GA, ACO and PSO solutions provide on average 3.92%, 4.02% and 3.90% saving in 340s, 311s and 329s, and there is not any UB reference. Average savings of S500 by the three meta-heuristics are 5.38%,



(A) Instances with 10, 20 and 50 vehicles



(B) Instances with 100, 200 and 500 vehicles

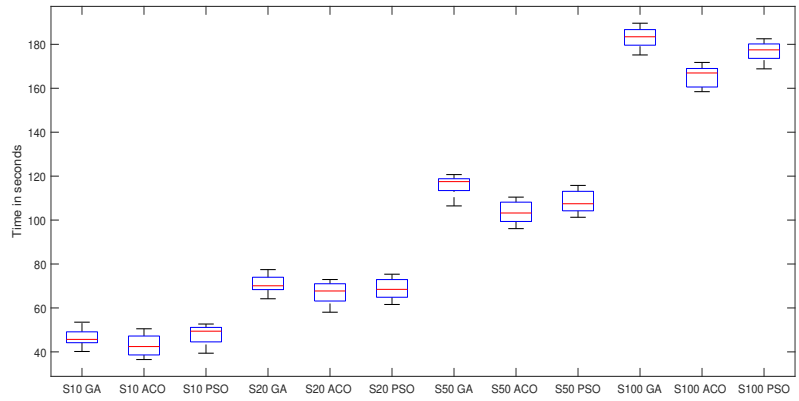


(C) Instances with 1000, 2000 vehicles

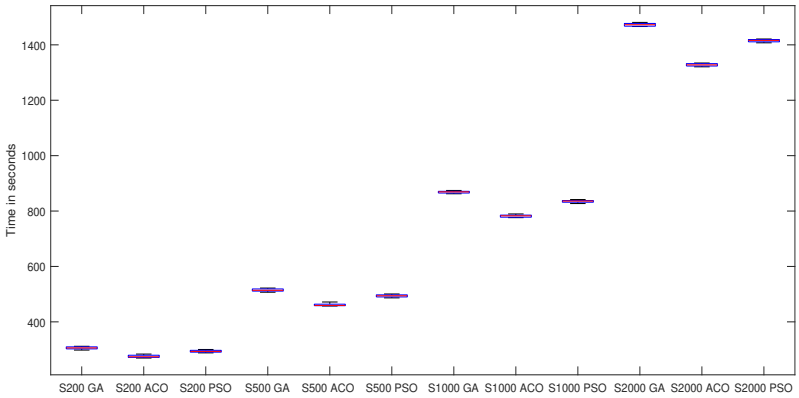
FIGURE 6.5: Upper bounds (UB) and savings with the three meta-heuristics GA, ACO and PSO on the Chicago network

5.46% and 5.33% obtained in 579s, 519s and 554s on average. S1000 examples are solved by GA, ACO and PSO giving 6.45% in 977s, 6.56% in 880s and 6.56% in 937s on average, respectively. The savings of the three methods corresponding to the S2000 instances are 7.38%, 7.56% and 7.34% obtained in 1657s, 1491s and 1589s.

The box plots of savings and times corresponding to the Swedish network are depicted in Figures 6.9 and 6.10. The results' summary is shown in Table 6.7. In



(A) Instances with 10, 20, 50 and 100 vehicles



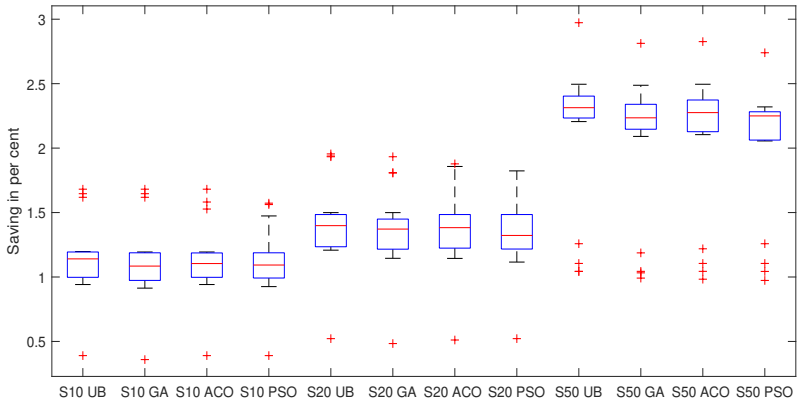
(B) Instances with 200, 500, 1000 and 2000 vehicles

FIGURE 6.6: Execution time of solving instances on the Chicago network with the three meta-heuristics of GA, ACO and PSO

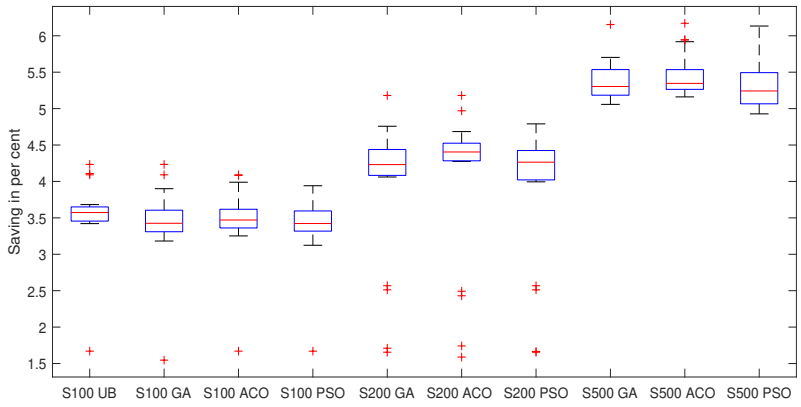
TABLE 6.5: The average savings, solution times and optimality gaps for the German network with the meta-heuristic methods

Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	GA	ACO	PSO	GA	ACO	PSO	GA	ACO	PSO
S10	1.21	1.21	1.20	46	43	48	0.03	0.04	0.03
S20	1.37	1.39	1.37	71	67	69	0.04	0.02	0.04
S50	2.19	2.22	2.19	116	103	108	0.07	0.04	0.07
S100	3.34	3.39	3.30	183	165	177	0.19*	0.14*	0.23*
S200	4.07	4.11	4.05	306	276	294	-	-	-
S500	5.57	5.66	5.49	515	461	494	-	-	-
S1000	6.83	6.97	6.72	869	835	783	-	-	-
S2000	7.67	7.81	7.57	1473	1328	1415	-	-	-

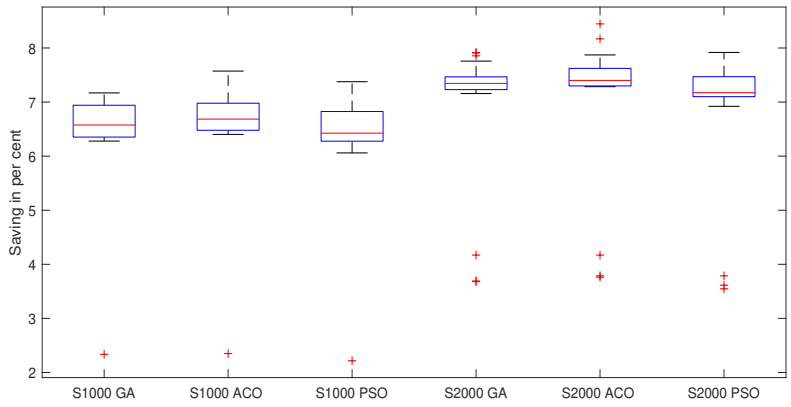
S10 cases, the average savings are 0.93%, 0.94% and 0.92%, the average execution times are 59s, 54s and 57s, and the mean optimality gaps are 0.02%, 0.01% and 0.03% for GA, ACO and PSO, respectively. In S20 cases, we obtain the average savings of 1.09%, 1.10% and 1.08% by GA, ACO and PSO. These results that are 0.03%, 0.02% and 0.04% distant from optimality, are obtained on average in 92s, 83s



(A) Instances with 10, 20 and 50 vehicles



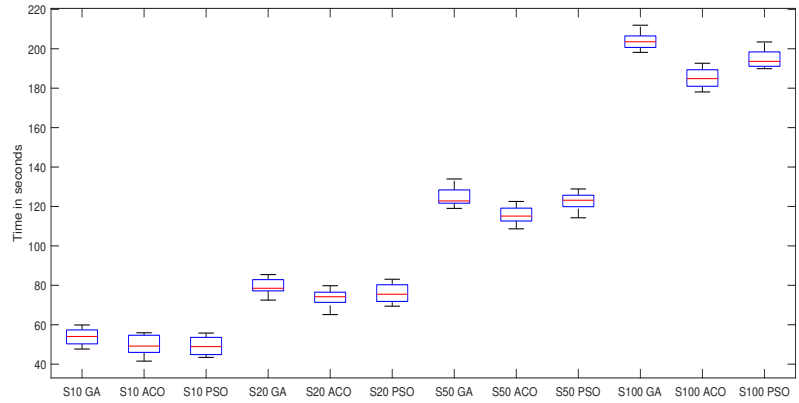
(B) Instances with 100, 200 and 500 vehicles



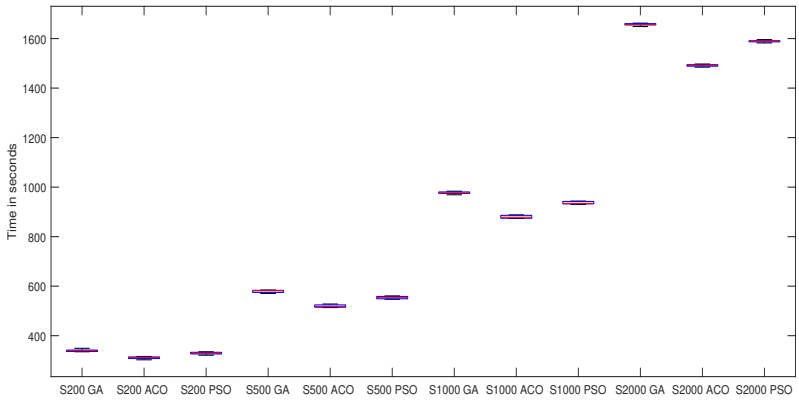
(C) Instances with 1000, 2000 vehicles

FIGURE 6.7: Upper bounds (UB) and savings with the three meta-heuristics of GA, ACO and PSO on the German network

and 87s. For S50, the savings of the three meta-heuristics amount to 1.90%, 1.94% and 1.88%, the optimality gaps are 0.07%, 0.03% and 0.09%, and these are obtained on average after 146s, 136s and 140s. Considering the results of S100 instances, we have attained 3.24%, 3.28% and 3.20% average saving in 236s, 217s and 227s average execution time and the results are averagely 0.12%, 0.08% and 0.16% away



(A) Instances with 10, 20, 50 and 100 vehicles



(B) Instances with 200, 500, 1000 and 2000 vehicles

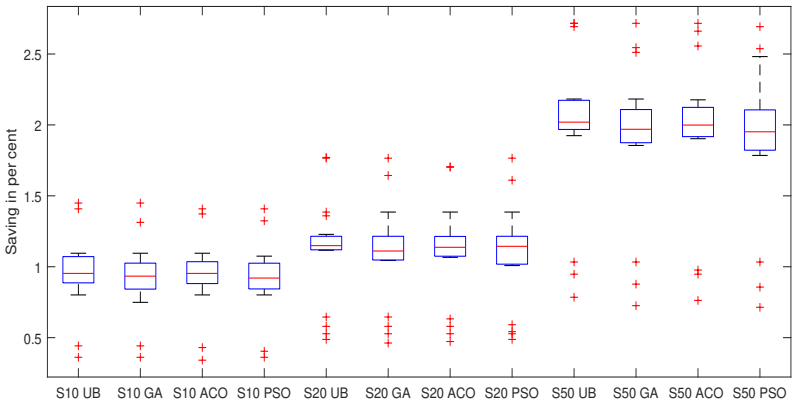
FIGURE 6.8: Execution time of solving instances on the German network with the three meta-heuristics GA, ACO and PSO

TABLE 6.6: The average savings, solution times and optimality gaps for the German network with the meta-heuristic methods

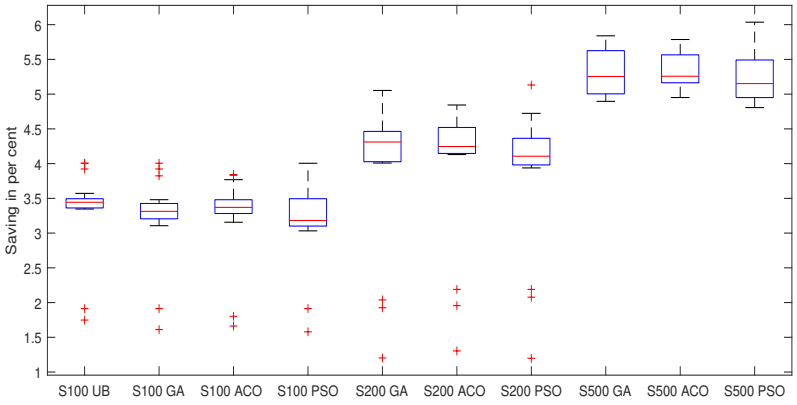
Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	GA	ACO	PSO	GA	ACO	PSO	GA	ACO	PSO
S10	1.12	1.12	1.10	54	49	49	0.02	0.02	0.04
S20	1.36	1.38	1.35	79	74	76	0.04	0.02	0.05
S50	1.99	2.01	1.85	125	116	123	0.14	0.12	0.28
S100	3.42	3.46	3.39	204	185	195	0.05*	0.01*	0.08*
S200	3.92	4.02	3.90	340	311	329	-	-	-
S500	5.38	5.46	5.33	579	519	554	-	-	-
S1000	6.45	6.56	6.56	977	880	937	-	-	-
S2000	7.38	7.56	7.34	1657	1491	1589	-	-	-

from the optimality. For S200, after elapsing the average computational times of 397s, 356s and 384s by GA, ACO and PSO, the savings corresponding to the three meta-heuristics are 3.97%, 4.01% and 3.89%, respectively. The average of best found solutions for S500 are 5.31%, 5.35% and 5.21% obtained in 672s, 606s and 645s. For S1000 samples, average savings of 6.12%, 6.22% and 6.05% are obtained in 1137s,

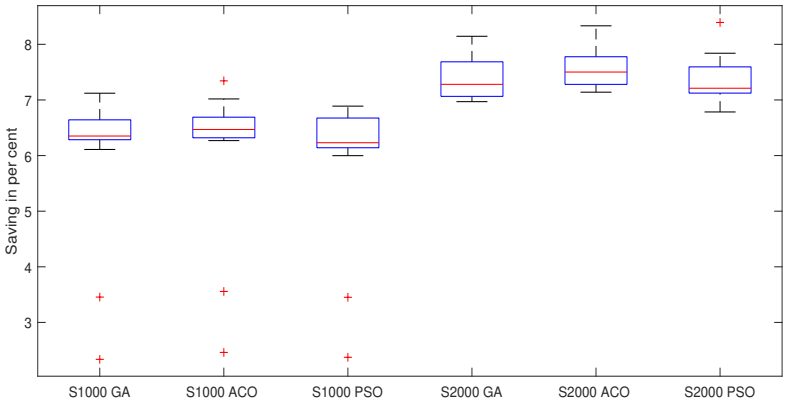
1026s and 1093s by GA, ACO and PSO. Finally, for S2000, the best saving found by the meta-heuristics are 6.92%, 7.02% and 6.79%, after exceeding the time limit of 1800s.



(A) Instances with 10, 20 and 50 vehicles



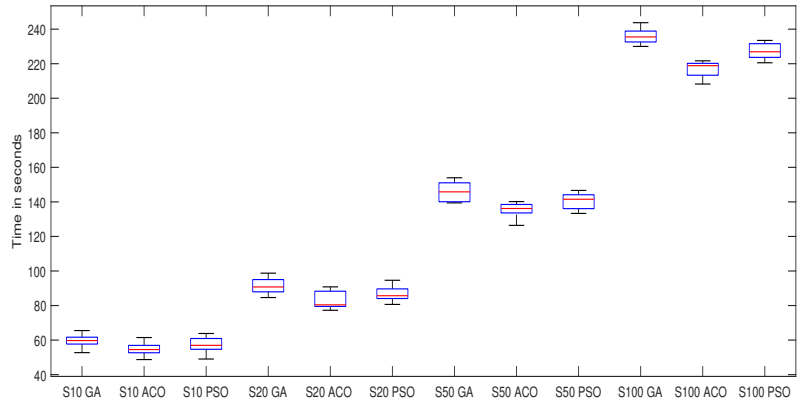
(B) Instances with 100, 200 and 500 vehicles



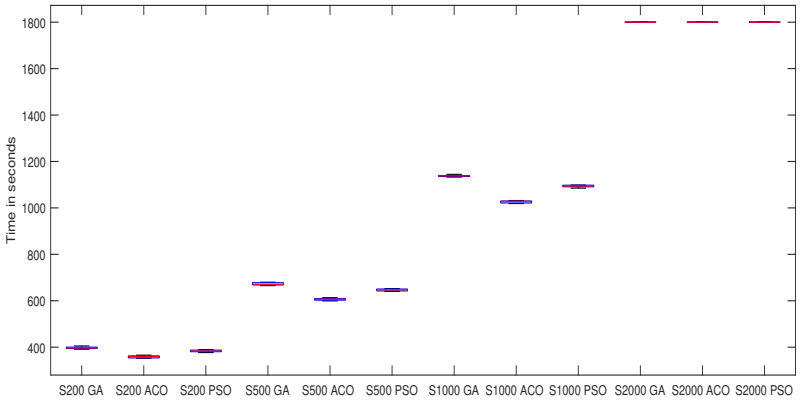
(C) Instances with 1000, 2000 vehicles

FIGURE 6.9: Upper bounds (UB) and savings with the three meta-heuristics of GA, ACO and PSO on the Swedish network

Figures 5.10 and 5.11 are dedicated to the savings and computation times of the instances on the Grid10 network and their averages are shown in Table 6.8. For



(A) Instances with 10, 20, 50 and 100 vehicles



(B) Instances with 200, 500, 1000 and 2000 vehicles

FIGURE 6.10: Execution time of solving instances on the Swedish network with the three meta-heuristics GA, ACO and PSO

TABLE 6.7: The average savings, solution times and optimality gaps for the Swedish network with the meta-heuristic methods

Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	GA	ACO	PSO	GA	ACO	PSO	GA	ACO	PSO
S10	0.93	0.94	0.92	59	54	57	0.02	0.01	0.03
S20	1.09	1.10	1.08	92	83	87	0.03	0.02	0.04
S50	1.90	1.94	1.88	146	136	140	0.07	0.03	0.09
S100	3.24	3.28	3.20	236	217	227	0.12*	0.08*	0.16*
S200	3.97	4.01	3.89	397	356	384	-	-	-
S500	5.31	5.35	5.21	672	606	645	-	-	-
S1000	6.12	6.22	6.05	1137	1026	1093	-	-	-
S2000	6.92	7.02	6.79	1800	1800	1800	-	-	-

S10, good average savings of 1.27%, 1.27%, 1.26% are achieved with GA, ACO and PSO methods. The savings are corresponding to the average gaps of 0.03%, 0.03% and 0.04% from optimality and are obtained averagely in 64s, 59s and 62s. For S20, the average savings are 1.62%, 1.63% and 1.60%, the average optimality gaps are 0.04%, 0.03% and 0.06% and the average execution times are 97s, 89s and 94s. For

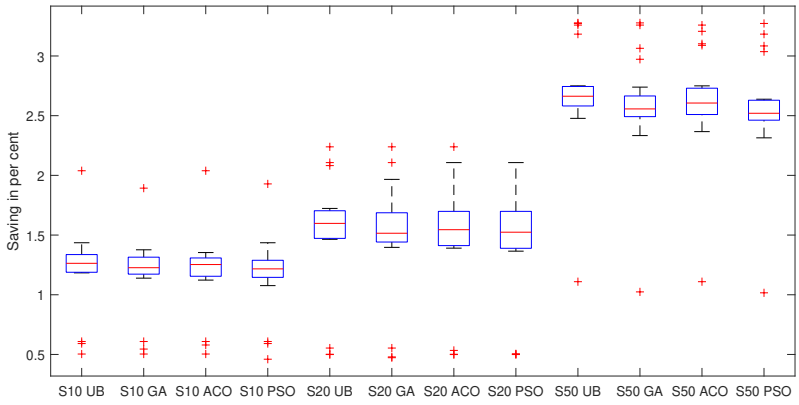
S50, the mean savings of the meta-heuristics are 2.57%, 2.62% and 2.56% obtained in average times of 156s, 143s and 150s. The savings are on average 0.11%, 0.06% and 0.12% distant from the optimal ones. The solutions of S100 samples by the three meta-heuristics give the average savings of 3.38%, 3.45% and 3.35% that are 0.24%, 0.17% and 0.27% away from the upper bounds and are reached after averagely 254s, 229s and 243s being elapsed. In case of S200, the average savings and times of the meta-heuristics are 4.71%, 4.79% and 4.64%, in 426s, 385s and 409s. We do not have any upper bound references. The meta-heuristics' provide for the S500 samples on average 5.53%, 5.65% and 5.50% saving in 721s, 650s and 693s. In dealing of GA, ACO and PSO with S1000 instances, 6.83%, 6.97% and 6.75% saving are achieved in 1220s, 1099s and 1723s. Finally, for the S2000, none of the algorithms reaches its maximum iteration in any case within the time limit. However, the best savings are 7.29%, 7.45% and 7.21% by GA, ACO and PSO, respectively.

TABLE 6.8: The average savings, solution times and optimality gaps for the Grid10 network with the meta-heuristic methods

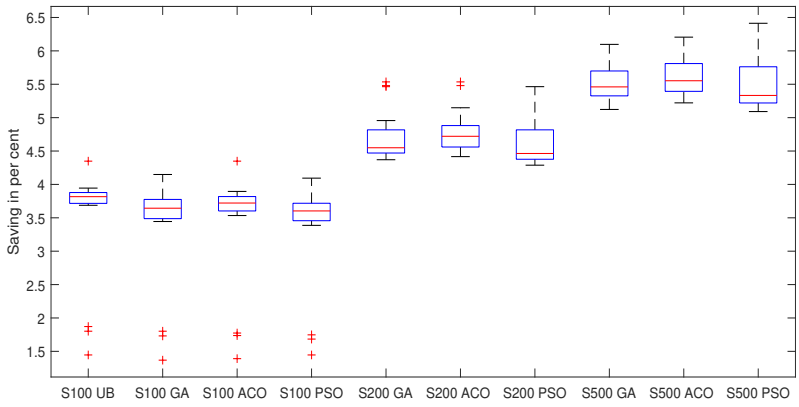
Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	GA	ACO	PSO	GA	ACO	PSO	GA	ACO	PSO
S10	1.27	1.27	1.26	64	59	62	0.03	0.03	0.04
S20	1.62	1.63	1.60	97	89	94	0.04	0.03	0.06
S50	2.57	2.62	2.56	156	143	150	0.11	0.06	0.12
S100	3.38	3.45	3.35	254	229	243	0.24*	0.17*	0.27*
S200	4.71	4.79	4.64	426	385	409	-	-	-
S500	5.53	5.65	5.50	721	650	693	-	-	-
S1000	6.83	6.97	6.75	1220	1099	1723	-	-	-
S2000	7.29	7.45	7.21	1800	1800	1800	-	-	-

Figures 5.12 and 5.13 include the box plots of savings and times resulted from applying GA, ACO and PSO algorithms to the samples on Grid30 network. The averages are given in Table 6.9. S10 solutions by the three meta-heuristics show average savings of 1.20%, 1.22% and 1.20% corresponding to the optimality gaps of 0.04%, 0.02% and 0.04%. These results are achieved after 73s, 67s and 73s of average computational time. For S20, the average savings of the methods are 1.47%, 1.48% and 1.46%, their average times are 116s, 106s and 112s, and their average optimality gaps are 0.04%, 0.03% and 0.05%. In solving S50 instances, we obtain 2.46%, 2.50% and 2.43% average saving in mean times of 186s, 170s and 179s with 0.10%, 0.06% and 0.13% average gap from optimality. For S100, after elapsing 304s, 275s and 291s, we locate solutions with average savings of 3.27% with GA, 3.36% with ACO and 3.25% with PSO. The average gaps from the upper bounds for the three algorithms are 0.18%, 0.09% and 0.20%, respectively. In terms of S200 examples, the obtained averages of saving are 4.48%, 4.54% and 4.41%, and average times are 511s, 461s and 490s corresponding to the three methods, respectively. For S500, the best found average saving of the 3 meta-heuristics are 5.22%, 5.34% and 5.18% obtained in 863s, 777s and 829s. GA, ACO and PSO give 6.38%, 6.50% and 6.30% average saving in 1464s, 1320s and 1404s of average time for S1000, while for S2000, these algorithms do not terminate and return 7.32%, 7.42% and 7.22% of average savings .

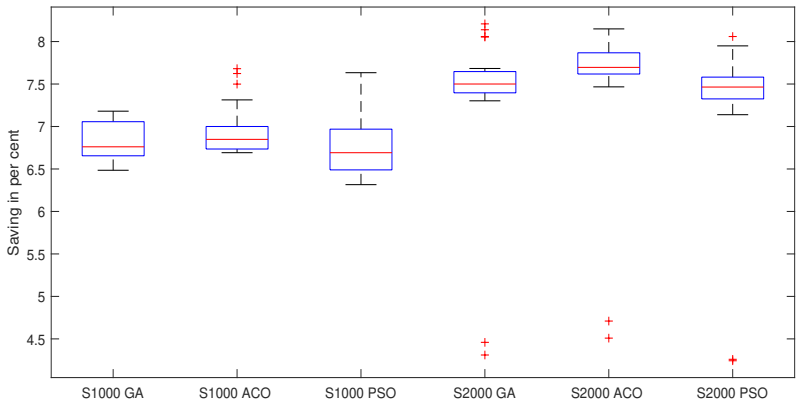
The last network and the hardest one is Grid50. The box plots of the results on this network are shown in Figures 6.15 and 6.16, and the averages are presented in Table 6.10. For S10, the average savings of the three meta-heuristic methods are 1.18%, 1.19% and 1.17%, their average solution times are 86s, 78s and 80s. The corresponding optimality gaps are 0.03%, 0.02% and 0.04%. For S20 examples, the average savings of GA, ACO and PSO are 1.29%, 1.30% and 1.28% in average times of 132s, 119s and 128s with average optimality gaps equal to 0.04%, 0.03% and 0.05%, respectively. Solving S50 samples with our three meta-heuristics, average savings of 2.39%, 2.42% and 2.35% are obtained in 214s, 196s and 203s on average, which are averagely 0.09%, 0.06% and 0.13% distant from the optimal solutions. For



(A) Instances with 10, 20 and 50 vehicles



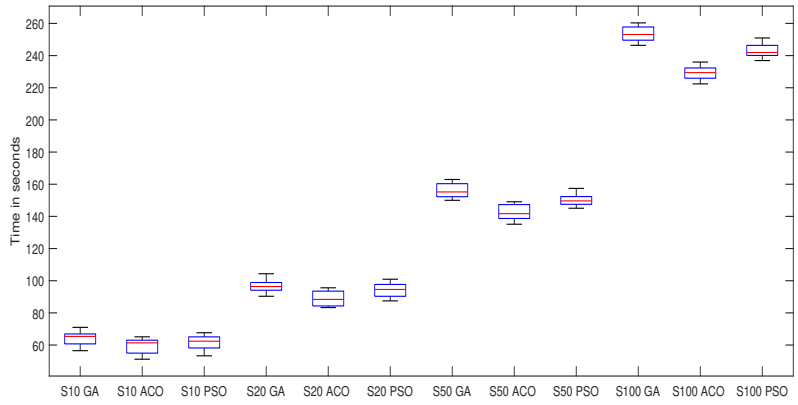
(B) Instances with 100, 200 and 500 vehicles



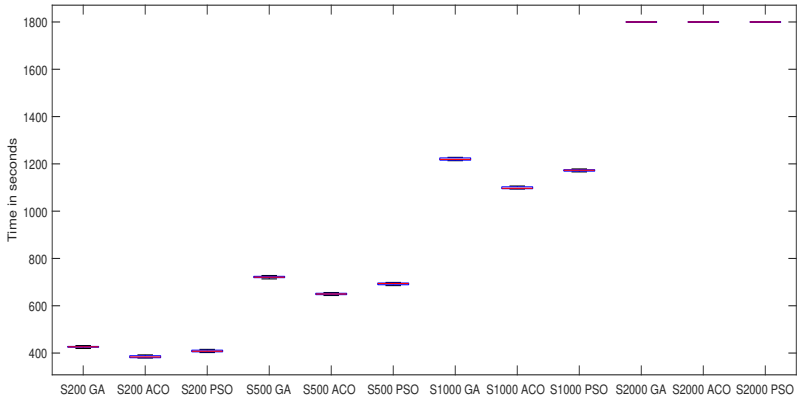
(C) Instances with 1000, 2000 vehicles

FIGURE 6.11: Upper bounds (UB) and savings with the three meta-heuristics of GA, ACO and PSO on the Grid10 network

S100, the average savings are 3.21%, 3.27% and 3.17%, the mean times are 350s, 317s and 336s, and the average gaps from UBs are 0.13%, 0.07% and 0.17% with the three meta-heuristics. The results of S200 samples show mean savings of 4.07%, 4.14% and 4.02% obtained in 590s, 533s and 564s by GA, ACO and PSO, respectively. For S500 samples, 4.77%, 4.84% and 4.72% of average saving are attained after 997s, 899s and 957s by the approaches. The methods give an average saving equal to



(A) Instances with 10, 20, 50 and 100 vehicles



(B) Instances with 200, 500, 1000 and 2000 vehicles

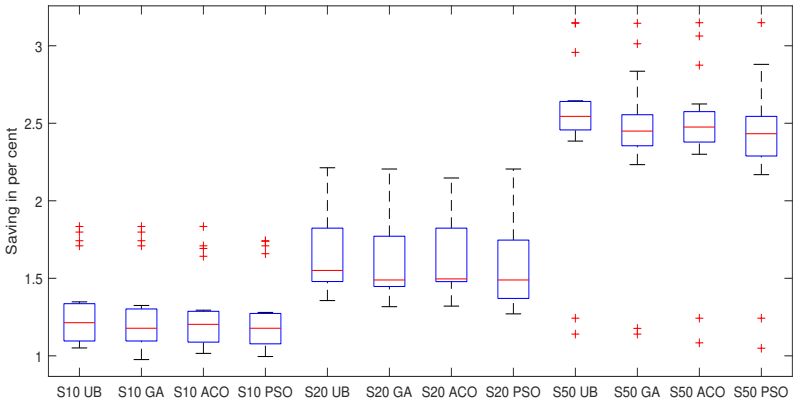
FIGURE 6.12: Execution time of solving instances on the Grid10 network with the three meta-heuristics GA, ACO and PSO

TABLE 6.9: The average savings, solution times and optimality gaps for the Grid30 network with the meta-heuristic methods

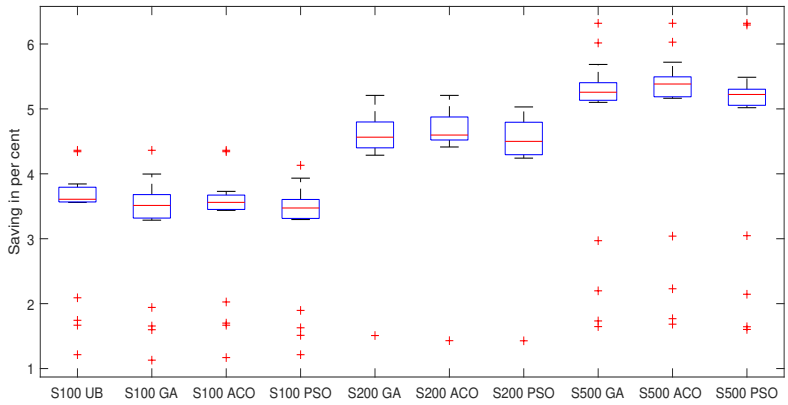
Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	GA	ACO	PSO	GA	ACO	PSO	GA	ACO	PSO
S10	1.20	1.22	1.20	73	67	73	0.04	0.02	0.04
S20	1.47	1.48	1.46	116	106	112	0.04	0.03	0.05
S50	2.46	2.50	2.43	186	170	179	0.10	0.06	0.13
S100	3.27	3.36	3.25	304	275	291	0.18*	0.09*	0.20*
S200	4.48	4.54	4.41	511	461	490	-	-	-
S500	5.22	5.34	5.18	863	777	829	-	-	-
S1000	6.38	6.50	6.30	1464	1320	1404	-	-	-
S2000	7.32	7.42	7.22	1800	1800	1800	-	-	-

5.95%, 6.06% and 5.86% in 1690s, 1520s and 1620s for S1000 samples, and finally, for S2000 they cannot terminate within our time limit but return 6.83%, 6.97% and 6.73% of average saving.

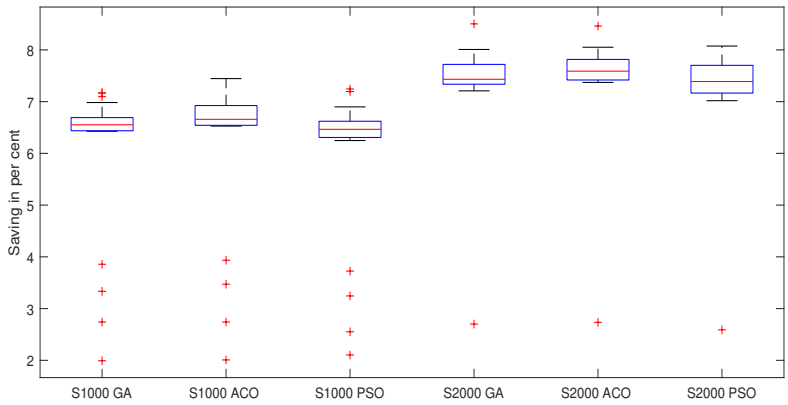
In an overall interpretation of the large volume of output data, it can be stated that the ACO method is able to achieve higher average savings in shorter average



(A) Instances with 10, 20 and 50 vehicles



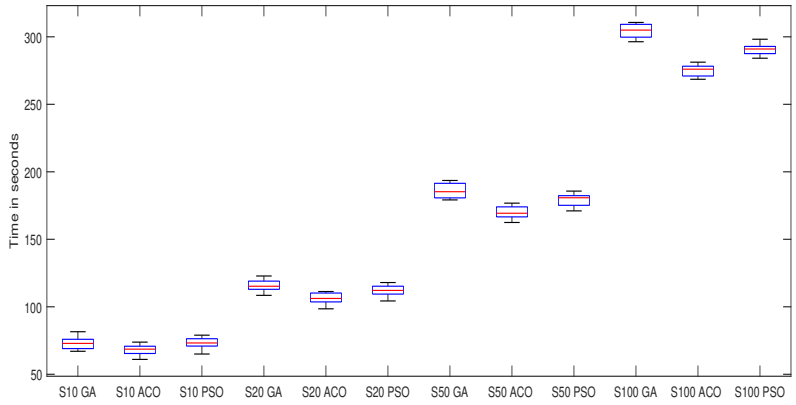
(B) Instances with 100, 200 and 500 vehicles



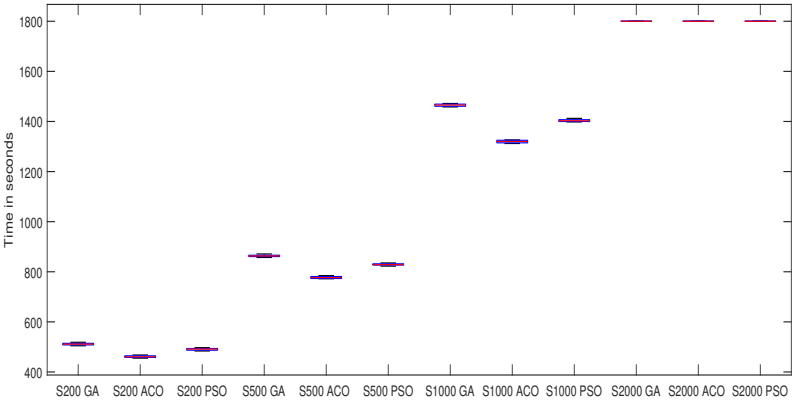
(C) Instances with 1000, 2000 vehicles

FIGURE 6.13: Upper bounds (UB) and savings with the three meta-heuristics of GA, ACO and PSO on the Grid30 network

times in comparison with GA and PSO. Regarding the GA and PSO performance, in most of the cases, GA savings are slightly better, whereas in terms of solution time, PSO is considerably superior. The reason is that our ACO is a novel approach that constructs solutions step by step and do not require any change (repair) in solutions to make them feasible. On the other hand, GA do not have such advantages and may need a solution repair after doing the crossover. While PSO works with a



(A) Instances with 10, 20, 50 and 100 vehicles



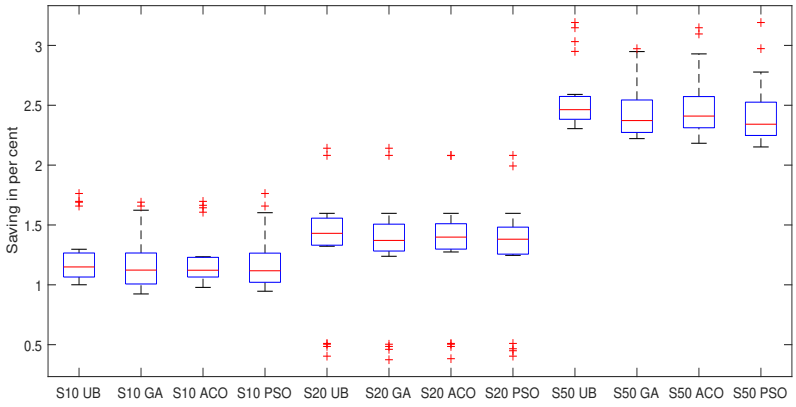
(B) Instances with 200, 500, 1000 and 2000 vehicles

FIGURE 6.14: Execution time of solving instances on the Grid30 network with the three meta-heuristics GA, ACO and PSO

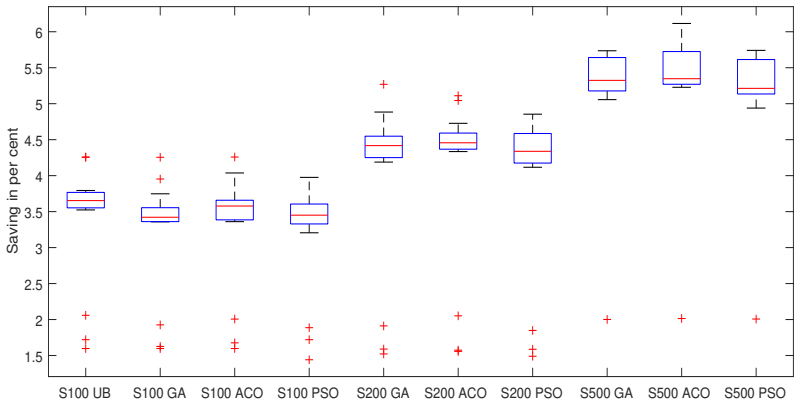
TABLE 6.10: The average savings, solution times and optimality gaps for the Grid50 network with the meta-heuristic methods

Problem Size	Avr. Saving (%)			Avr. Time (s)			Avr. Optimality Gap or Avr. Gap from UB* (%)		
	GA	ACO	PSO	GA	ACO	PSO	GA	ACO	PSO
S10	1.18	1.19	1.17	86	78	80	0.03	0.02	0.04
S20	1.29	1.30	1.28	132	119	128	0.04	0.03	0.05
S50	2.39	2.42	2.35	214	196	203	0.09	0.06	0.13
S100	3.21	3.27	3.17	350	317	336	0.13*	0.07*	0.17*
S200	4.07	4.14	4.02	590	533	564	-	-	-
S500	4.77	4.84	4.72	997	899	957	-	-	-
S1000	5.95	6.06	5.86	1690	1520	1620	-	-	-
S2000	6.83	6.97	6.73	1800	1800	1800	-	-	-

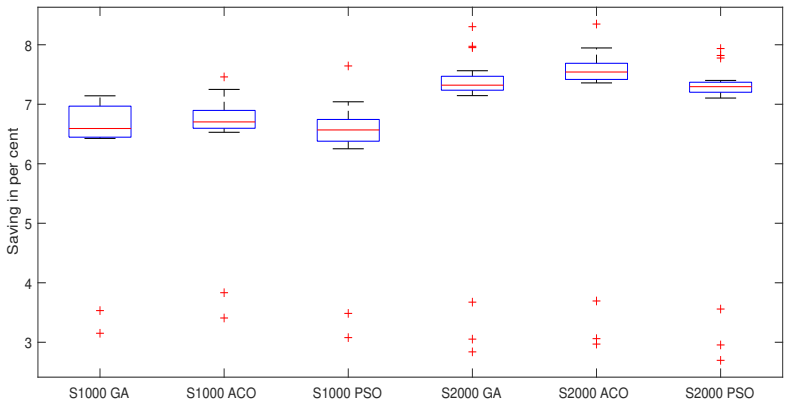
continuous solution domain and our problem solutions have discrete elements, it can achieve good results, which are only in some cases marginally lower than the results of GA. Moreover, any new solution achieved by the PSO scheme is also feasible and do not require any repairing. This accelerates the algorithm. Furthermore, it needs the tuning of only 2 parameters, which requires less effort



(A) Instances with 10, 20 and 50 vehicles



(B) Instances with 100, 200 and 500 vehicles



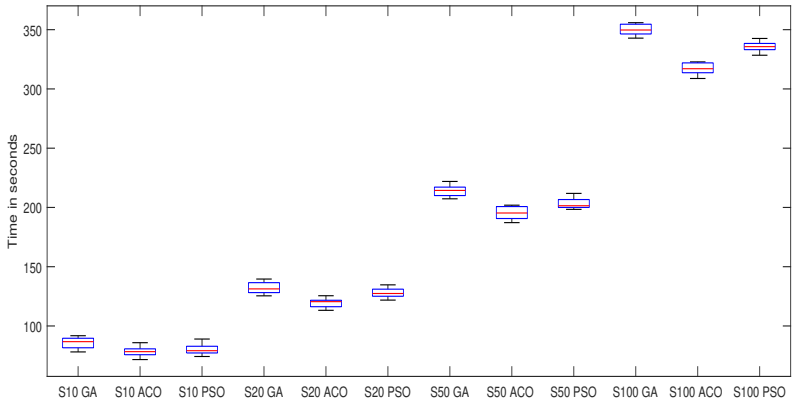
(C) Instances with 1000, 2000 vehicles

FIGURE 6.15: Upper bounds (UB) and savings with the three meta-heuristics of GA, ACO and PSO on the Grid50 network

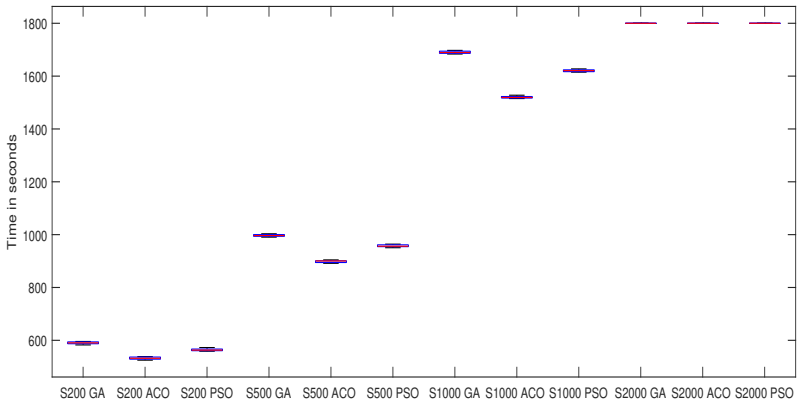
compared to the other two meta-heuristics.

The overall averages of savings and times obtained on all the road networks are graphically shown in Figure 6.17 and 6.18. They display again the mentioned conclusions about the performance of our meta-heuristics.

To show how the three meta-heuristics perform from the first to last iteration, two single instances of S1000 and S2000 on the Chicago network are chosen and the



(A) Instances with 10, 20, 50 and 100 vehicles



(B) Instances with 200, 500, 1000 and 2000 vehicles

FIGURE 6.16: Execution time of solving instances on the Grid50 network with the three meta-heuristics GA, ACO and PSO

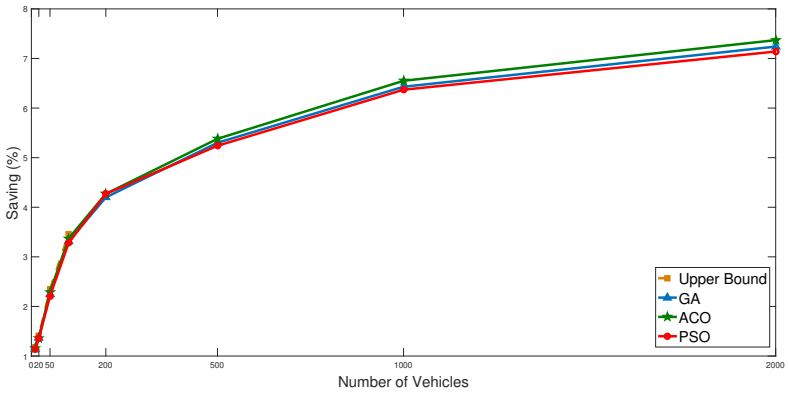


FIGURE 6.17: The overall average of the savings provided by the 3 meta-heuristics regarding all the 6 road networks

best results of iterations are drawn in Figure 6.19. Unlike the previous results, here we show the objective function values of iterations instead of the saving percentages. These objective values are obtained by equation 3.21 of Chapter 3. Therefore, lower

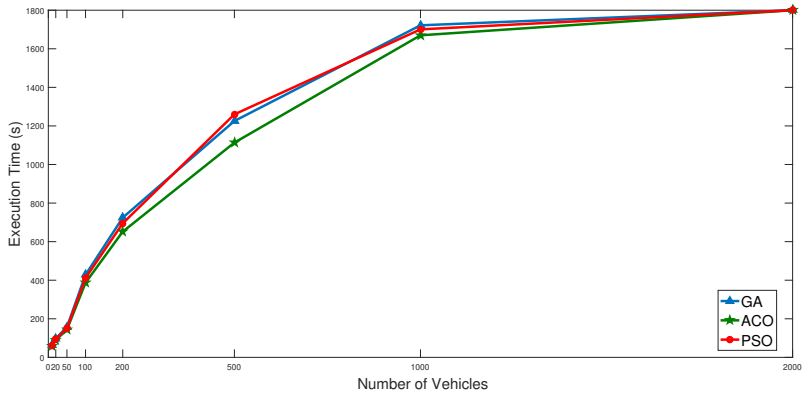
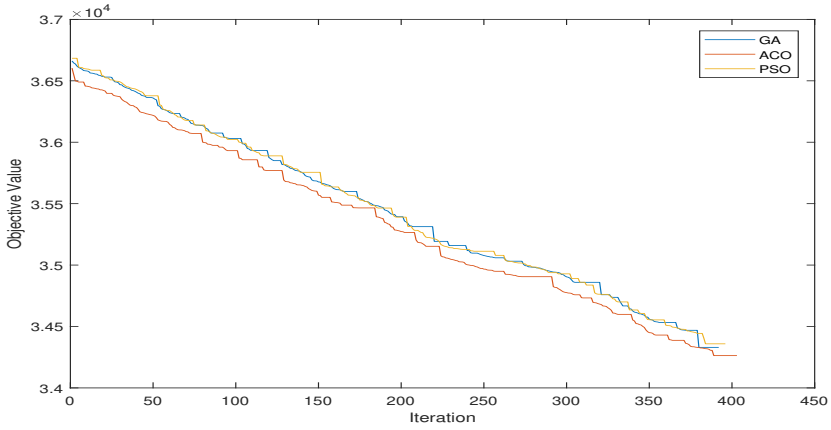
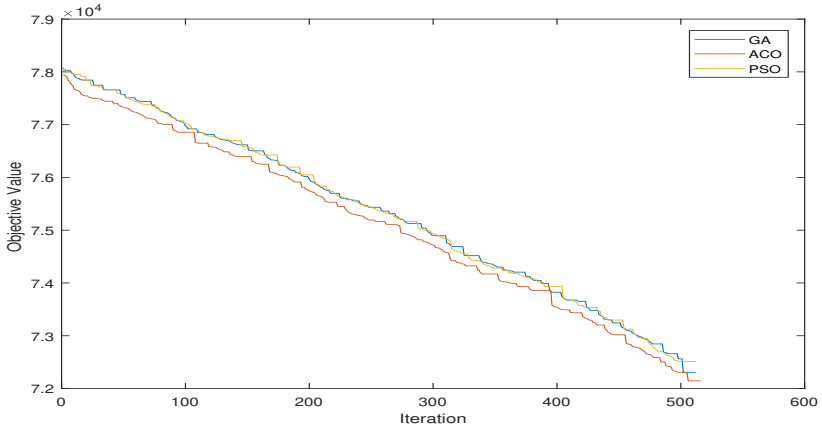


FIGURE 6.18: The overall average of the solution times of the 3 meta-heuristics regarding all the 6 road networks

objective values are better. As it can be seen, the results of ACO are most of the times under those of GA and PSO that shows the superiority of this method.



(A) An instance of S1000



(B) An instance of S2000

FIGURE 6.19: The best results of GA, ACO and PSO in each iteration (convergence plots) for a single instance on the Chicago network

Usually very short solution times are expected from meta-heuristics, nonetheless, it is observed that the times are too much longer in dealing with the FEP problem and specially for the big sizes. This makes our meta-heuristics unable to reach their final iteration, which is determined for them in the tuning of Section 6.4, within 1800s in many cases of S2000. The reason behind these strange long computational times is not related to the meta-heuristics' schemes, rather it is about the complicated procedure that is required to evaluate each solution. This evaluation procedure entails, firstly, converting the encoded solutions into routings, and then, running a couple of algorithms from Chapter 5 to do the time scheduling, speed adjustment and calculate the objective value. This significantly slows down each iteration of any meta-heuristic.

Nevertheless, the meta-heuristics have shown good capability in terms of both solution quality and time, if they are compared to the exact and heuristic methods introduced in the previous chapters. Their average savings are considerably higher and their computation time is much shorter than the best heuristic option for each size. Hence, it can be concluded that the best alternatives to deal with the FEP problem in realistic large sizes are meta-heuristics and among the three examined approaches in this category, ACO has the best performance. This is also statistically proved in Section 6.6.

6.6 Statistical Tests

Here non-parameteric statistical tests of Friedman with Bergmann-Hommels post-hoc procedure are conducted to compare the savings and execution times among the three meta-heuristics, and also (almost) the best savings of heuristics obtained by Global Planning (GP) and the shortest time of heuristics related to Hub (H) method. The obtained p-values of comparisons are shown separately for savings and times in Tables 6.11 and 6.12, respectively.

Comparison \ Size	S10	S20	S50	S100
GA vs. ACO	1.5404e-01	0.0100	3.6144e-04	7.9662e-05
GA vs. PSO	0.1269	1.6808e-01	1.1996e-02	6.8656e-03
ACO vs. PSO	0.0095	2.3173e-04	1.1788e-09	2.8925e-11
GA vs. GP	3.5295e-06	7.3799e-09	3.3263e-07	1.1424e-09
ACO vs. GP	1.9676e-09	0.0000e+00	0.0000e+00	0.0000e+00
PSO vs. GP	1.7187e-03	9.0499e-06	5.1792e-03	3.7312e-04

Comparison \ Size	S200	S500	S1000	S2000
GA vs. ACO	1.4586e-04	0.0495	6.9651e-09	1.8722e-11
GA vs. PSO	8.8044e-04	0.0961	1.6700e-05	1.5811e-05
ACO vs. PSO	9.1127e-13	0.0009	0.0000e+00	0.0000e+00
GA vs. GP	1.2181e-09	0	-	-
ACO vs. GP	0.0000e+00	0	-	-
PSO vs. GP	3.4390e-03	0	-	-

TABLE 6.11: The p-values of the Friedman test with Bergmann-Hommel post-hoc procedures for the pairwise comparisons of GA, ACO and PSO savings, and the best saving among heuristics obtained by GP

Since the problems are solved by heuristics up to S200 within the time limit, their comparisons with meta-heuristics are done for the sizes from S10 to S500 in terms of saving and to S200 in terms of time. As it is evident from the p-values, the GP, which is the best heuristic in terms of solution quality (the results of BP are quite the same in some cases), and H, which has the shortest execution time among the heuristics, cannot compete with any of the meta-heuristics in terms of saving and time. This is proved by the near zero p-values.

Comparison \ Size	S10	S20	S50	S100
GA vs. ACO	1.0352e-05	2.0669e-07	1.5721e-10	9.3259e-15
GA vs. PSO	0.0448	0.0078	0.0015	1.1053e-04
ACO vs. PSO	0.0084	0.0062	0.0007	5.8940e-05
GA vs. H	0	0	0	0
ACO vs. H	0	0	0	0
PSO vs. H	0	0	0	0

Comparison \ Size	S200	S500	S1000	S2000
GA vs. ACO	0.0000e+00	0	0	5.9952e-15
GA vs. PSO	3.7378e-05	0	0	3.2099e-01
ACO vs. PSO	1.0437e-05	0	0	3.7310e-12
GA vs. H	0	-	-	-
ACO vs. H	0	-	-	-
PSO vs. H	0	-	-	-

TABLE 6.12: The p-values of the Friedman test with Bergmann-Hommel post-hoc procedures for the pairwise comparisons of GA, ACO and PSO times, and the shortest solution time among heuristics obtained by H

Comparing the meta-heuristics, the low p-values show significance difference between the saving and time of the ACO with GA and PSO that verifies the superiority of the ACO. Except for S10, S20 and S500 for saving and S2000 for time, the GA and PSO performances are different in the significance level of 0.05.

Hence the methods can be sorted in terms of saving as ACO $\succ$ GA $\succ$ PSO $\succ$ GP and regarding the solution time as ACO $\succ$ PSO $\succ$ GA $\succ$ H from the best to the weakest.

6.7 Summary

In this chapter, three meta-heuristic solution methodologies of GA, ACO and PSO were presented, which find good FEP solutions by different schemes. They help us to increase the tractable scalability of the problem, so we could achieve excellent solutions for instances including up to 2000 vehicles (S2000). Appropriate solution encoding and decoding concepts were defined which use some algorithms of Chapter 5 for time scheduling, speed determination and solution evaluation. The results and statistical tests verify that the meta-heuristics are considerably superior to heuristics in terms of both the savings and computation time, and moreover, among the three meta-heuristics, ACO performs obviously better. Having these solution tools, large scale fuel efficient platooning instances which are analogous to real cases can be successfully tackled. The next chapter investigates the question that how much the changing of the input data of an FEP problem, that regularly happens in real life, can influence the final results.

Chapter 7

Sensitivity Analyses

So far, we have successfully solved the fuel efficient platooning problem of real scales and achieved good savings. However, the FEP instances are solved based on the inputs, which can easily change in real life. In this chapter, the sensitivity of final results to the change in the input parameters of the problem is investigated. The number of vehicles, unit saving factor that can be achieved by platooning, time constraints and speed profiles are the main parameters that the problem is solved with different settings of them. The organisation of this chapter is as follows: in Section 7.1, the effect of involving more vehicles, in Section 7.2, the influence of increasing the saving factor of platooning, and in Section 7.3, the effect of loosening the time restrictions for departing from the origin and arriving at the destination, are examined. Subsequently, Section 7.4 investigates how the results change if there is only one speed option which is the cruise speed. Section 7.5 analyses the advantage of having the permission to choose routes longer than the shortest path in FEP solutions. In Section 7.6, the individual savings that the participating vehicles experience in the system are discussed. Finally, Section 7.7 covers the conclusions of this chapter.

7.1 Increasing the Number of Vehicles

In this part, the effect of involving more vehicles in the system on their platooning behaviour and final obtainable benefit are researched. For this sake, a new measure is introduced which is directly proportional to saving and can better indicate the platooning behaviour of the whole system. This measure is called *Rate of Platooning (ROP)* and calculated as dividing the sum of distances that all vehicles drive in platoons by the total distance that they drive either in platoon or alone, and it is expressed in per cent:

$ROP = \frac{Distance\ travelled\ inplatoon}{Total\ distance\ travelled} \times 100$

ROP is used throughout this chapter besides the saving percentage because it shows the proportion of the total platooning distance, which is very practical in our sensitivity analyses.

To involve all the 6 networks used in this thesis, for each size, 5 different instances on each network are chosen. Therefore, there are 30 samples for each size in this analysis.

In the sensitivity analyses of this chapter, the average saving and ROP are calculated for problem sizes S10 to S2000, which are solved by the ACO method given by Algorithm 11. This is because the ACO has been proved to be the fastest method that provides excellent results in the previous chapter. The effect on the saving and ROP by additional vehicles are here addressed. They are depicted by the number of vehicles in Figure 7.1. Increasing the number of vehicles from 10 to 2000, we can grow the total distance driven in platoon from 12.18% to 77.61% resulting in corresponding savings from 1.11% to 7.52%.

As it can be seen, in the beginning, increasing the number of participants (vehicles) results in larger growth in the average saving and ROP. However, by going further, the rate of benefit from the added participants decreases. This fact is evident by the line slopes. The reason is that by having less vehicles in the road

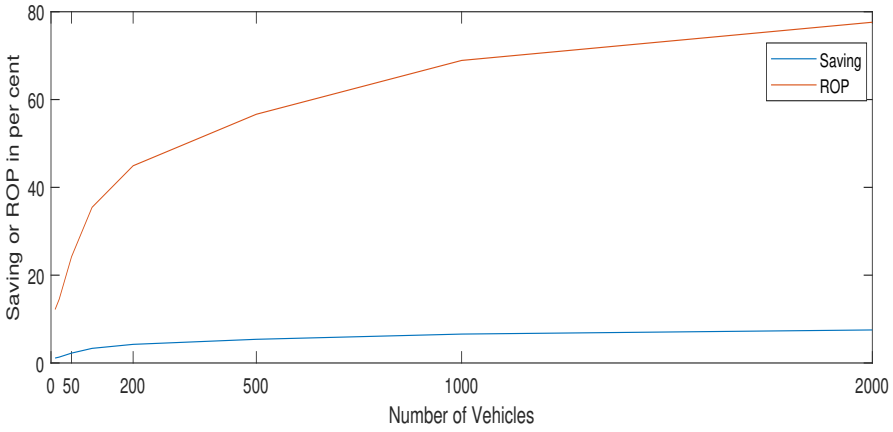


FIGURE 7.1: The average saving and ROP by different numbers of vehicles

network, there are lots of platooning potentials which can be used by the next added vehicles. On the other hand, by existing a large number of vehicles, the platooning opportunities are mostly used, and hence, the saving and ROP provided by extra participants become considerably less.

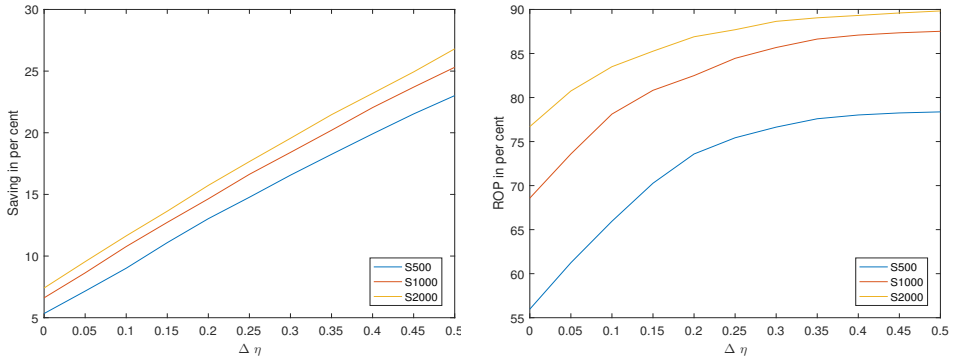
7.2 Increasing the Saving Factor of Platooning

In this part, the effect of increasing the attainable saving factor of platooning or η_s, which has been presented and used in the FEP models of Chapter 3, is analysed. η_s is the platooning incentive of the system. Although the large values of η_s may seem unrealistic, they can indeed be possible if besides the fuel saving other profits like drivers' wage reduction by autonomous driving of the following vehicles or road traffic alleviation are achievable through platooning.

Since there are three different speed profiles, three different values exist for η_s based on s in our work. Therefore, an amount called $\Delta\eta$ is added to the three η_s each time. We increase this $\Delta\eta$ and calculate the corresponding savings and ROPs based on $\eta_s + \Delta\eta$. As the maximum detour (MD^v) can act against the effect of the enlarged saving factor by prohibiting vehicles from choosing lager alternative routes, all MDs^v and their corresponding constraints, i.e. 3.18 or 3.35 in Chapter 3 are eliminated. This influences on the initial phase or finding the feasible routes (see Algorithm 1 in Chapter 5) in solving process by the ACO, which is used here.

These examinations are done for the three largest problem sizes with 500, 1000 and 2000 vehicles (S500, S1000 and S2000) with 5 instances on each network (in total 30 instances) and the related average saving and ROP plots are shown in Figure 7.2.

It is observable that by increasing η_s, the ROP and saving rise considerably. Increasing $\Delta\eta$ from 0 to 0.5, the average saving grows from 5.34% to 23.09% for S500, from 6.59% to 25.44% for S1000 and from 7.41% to 27.12% for S2000. The average ROP increases from 55.97% to 78.02%, from 68.58% to 87.09% and from 76.70% to 89.32% for S500, S1000 and S2000, respectively. The total saving considerably increases because, firstly, larger discounts are gained by the platooned vehicles. Besides, the total proportion of in platoon travelled distance is grown that results in more saving. At first, the rate of growth in ROP is larger and it gradually decreases. This is because, firstly, the platooning opportunities are mainly used, and secondly, the time constraints prevent vehicles from driving on very large routes that do not respect the deadlines.



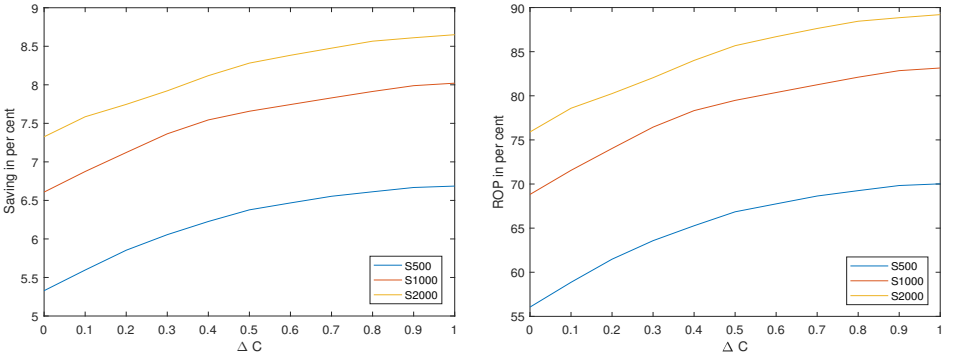
(A) Saving (B) ROP

FIGURE 7.2: The average saving and ROP by increasing the saving factor of platooning through $\Delta\eta$

7.3 Easing the Time Restrictions

In this part, the constraints related to the release times and deadlines of vehicles are gradually softened. In other words, we widen the travelling time windows of vehicles. For widening the time windows, we simultaneously deduct an amount from the earliest release times (T_e^v) and add another amount to the deadlines (T_{max}^v) both according to a factor called ΔC. The new release times and deadlines are calculated as: $T_e^v = T_e^v - \Delta C(T_e^a)$; $T_{max}^a = T_{max}^a + \Delta C(H - T_{max}^a)$. ΔC is increased from 0 to 1. ΔC=1 means that there is neither any time limit for departing from the origin nor for arriving at the destination. Like in the previous section, the maximum detours of vehicles (MD^v) are all ignored here as well.

The tests are done by 30 samples, 5 on each network, for each value of ΔC. Figure 7.3 illustrates the change in the average ROP and saving for the three sizes of S500, S1000, S2000 by easing time constraints through increasing ΔC. The growing of ROP and saving is up to approximately ΔC=0.9 and they do not noticeably increase afterwards. The reason is that η_{s_3}, which is equals to 0.1, acts in the opposite direction and prevents vehicles from choosing paths which are more than 10% longer than their shortest path(s) due to non-optimality. Like in the previous analyses and for the same reason, the ROP and saving growth are more with less number of vehicles.



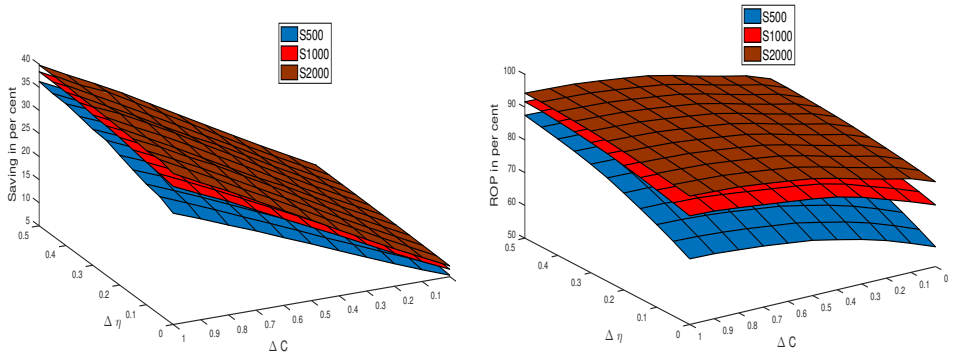
(A) Saving (B) ROP

FIGURE 7.3: The average saving and ROP by easing the time constraints by ΔC

As it is observed that both the saving factor and time windows prohibit the ROP from excessing an specific amount, here they are enlarged together simultaneously

to find out how much further progress on the ROP and saving can be attained. As the combinations of $\Delta\eta$ and ΔC are numerous and so many experiments are required, in this part, we employ only 1 example on each network. This means altogether 6 examples for each combination.

Figure 7.4 shows the results of increasing both $\Delta\eta$ and ΔC. It is evident that the average ROP and saving percentage can be grown to more than 92.51% and 39.84% with S2000 since none of the fuel saving factor and time constraints resists against the other one. Due to the increase in η_s, the same platooning operation provide higher savings. The ROP cannot be further increased noticeably because for the platooning coverage of the remaining distances, some vehicles must travel on the routes which are so long that even the largest $\Delta\eta$ of 0.5 can not make them attractive and selectable.



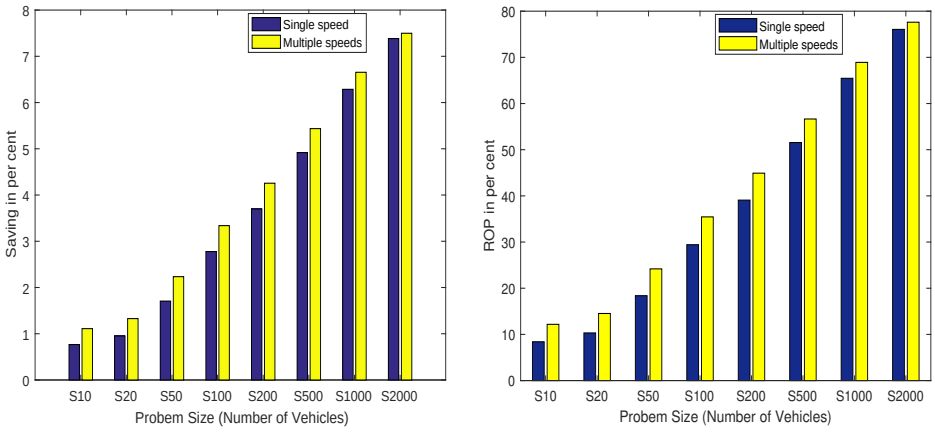
(A) Saving (B) ROP

FIGURE 7.4: The average saving and ROP by simultaneously increasing $\Delta\eta$ and easing the time constraints by ΔC

7.4 Effect of Multiple Speed Options

In this section, the advantage of the FEP model with 3 different speeds which are explained in Table 3.1 of Chapter 3 is investigated. Although the existence of multiple speeds enables acceleration of some vehicles to catch platoons, it adds more complexity to the problem. Hence, it is worthwhile to test the reduction in the saving and ROP in case that we have only one speed which is s_3 or cruise speed. For this examination, 5 different random instance on each network (altogether 30) for each size are solved once based on having a single speed and another time normally with the 3 speed profiles, then the average of results are compared. In solution process with ACO, this change affects Algorithm 2 and the embedded Algorithm 3 which are presented in Chapter 5.

The comparisons of solutions with single and our usual 3 speed options are shown for each size by the bar charts in Figure 7.5. The relative difference for saving and ROP are calculated as $1 - \frac{Saving(ROP)\ by\ single\ speed}{Saving\ (ROP)\ by\ multiple\ (3)\ speeds}$. For the 8 sizes S10 to S2000, the relative saving differences are 0.31, 0.28, 0.24, 0.17, 0.13, 0.10, 0.06 and 0.02, and the relative ROP differences are quite the same: 0.31, 0.29, 0.24, 0.17, 0.13, 0.09, 0.05 and 0.02. Based on these results, it can be stated that by more vehicles, the contribution of multiple speeds decreases. The reason is when a large number of vehicles are spread on the road network, they do not require to accelerate much in order to catch platoons because there are enough platooning opportunities nearby which can be exploited by the cruise speed.



(A) Saving (B) ROP

FIGURE 7.5: The comparison between the case that only one speed (cruise option) is considered in the model and the normal case of this thesis with multiple (three) speed profiles

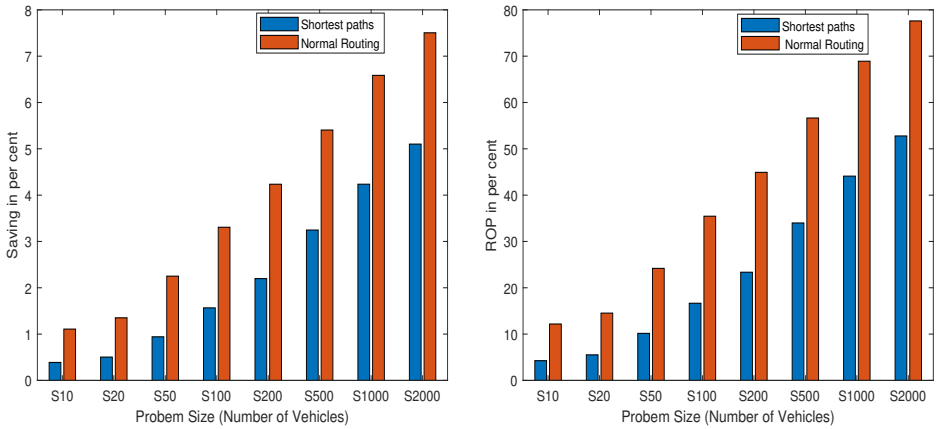
7.5 Shortest Path vs. Routing

The main part in solving an FEP problem is routing of vehicles which is provided by our solution methodologies. In the normal routing used so far in this thesis, the feasible or candidate routes are those directing vehicles from their origin to destination without violating any time or distance constraints, which are presented in our FEP models of Chapter 3. Another approach is allowing vehicles to only choose their shortest path(s), therefore, only the shortest paths are in the set of feasible routes of each vehicle and not any longer path. There can be multiple shortest paths for a vehicle that connect the staring to the ending point and as explained in Section 3.4.1 of Chapter 3, there are many alternative shortest paths between two nodes on Grid networks.

The analyses of this section focus on comparing the final saving and ROP of the routing only on shortest paths vs. normal routing. For this goal, 30 samples (5 on each network) for each size are solved based on the both approaches. Then, averages of saving and ROP are calculated. The average saving and ROP are compared by bar charts in Figure 7.6. The relative differences are calculated exactly like in the previous section as: $1 - \frac{Saving(ROP)\ by\ shortest\ paths}{Saving\ (ROP)\ by\ normal\ routing}$. Relative saving differences for S10 to S2000 are 0.65, 0.63, 0.58, 0.53, 0.48, 0.40, 0.36 and 0.32, respectively. Similarly, the relative ROP differences of the 8 sizes amount to 0.65, 0.62, 0.58, 0.53, 0.48, 0.40, 0.36 and 0.32. It is again self-evident that the investigation in routing by taking longer routes into account yields more benefits in smaller sizes as there are less vehicles in the network. Hence, exploring all of the possible routes is beneficial. Contrarily, by a large number of vehicles being scattered throughout the network, the importance of a complete search among routes decreases because enough platooning chances exist on the shortest paths.

7.6 Individual Benefit of Vehicles

In this section, the FEP problem is analysed from the aspect of participating vehicles. The saving and ROP experienced by each vehicle as a single member of the system is calculated. Here, it is assumed that the vehicle which arrives first at the joining point of the platoon takes on the role of leader. In case that there are more than one vehicle with the minimum arrival time at that point, the leader is chosen randomly among them. The saving of a single vehicle v is defined as the ratio:



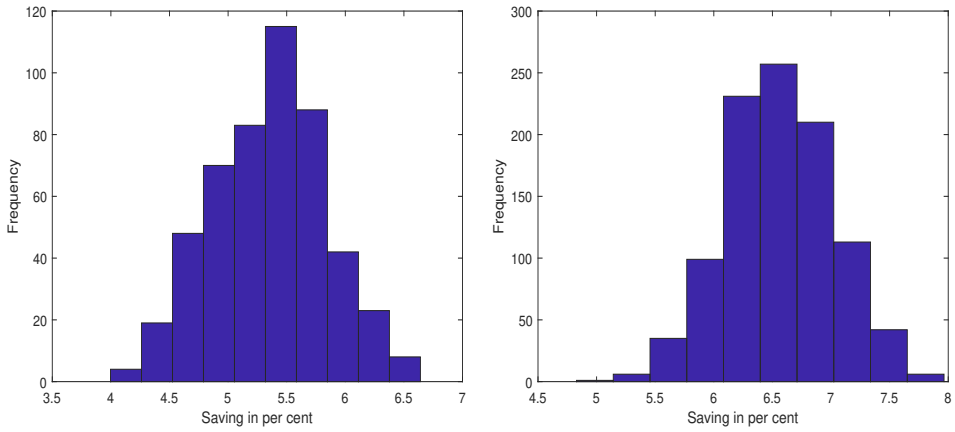
(A) Saving (B) ROP

FIGURE 7.6: The comparison between the case that the vehicles are only allowed to drive on their shortest paths vs. the normal routing.

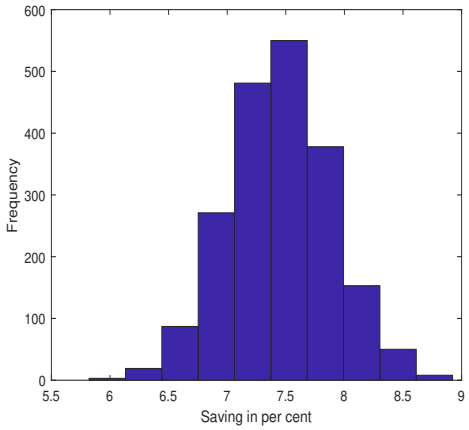
$\frac{Fuel\ cost\ of\ v\ in\ the\ platooning\ solution}{Fuel\ cost\ of\ v\ in\ the\ shortest\ path\ solution\ wtihtout\ platooning}$; and its ROP is: $\frac{The\ distance\ that\ v\ drives\ in\ platoon}{The\ total\ distance\ that\ v\ drives}$. For the 30 samples of S500, S1000 and S2000, which are equally divided between the 6 networks, the average saving and ROP are calculated for the participants. The histograms showing the frequencies of average savings and ROP values among the vehicles are presented as Figure 7.7 and 7.8, respectively. As it can be seen in the histograms, all vehicles benefit from platooning. Some vehicles experience less saving because their ROP is also less. Nonetheless, it can happen that even with a good ROP, the saving is not considerable because the vehicle drives a large part of its path as the leader of platoons, and therefore, it cannot benefit from any saving on those routes. The frequencies of observations around the averages are high as expected.

7.7 Summary

In this chapter, some important experiments were conducted that aimed mostly at finding the relationship between some important inputs of the FEP problem and the final results. This is crucial because these inputs may change by time or when innovations are applied to the transportation systems. On the other hand, these sensitivity analyses help us to understand that the change of which inputs can provide more increase in the platooning benefit. At the end of this chapter, it was shown that in our platooning samples, all of the participants attain on average a saving through several participation . This can be considered as an important individual incentive for each vehicle to attend the platooning program apart from the outcomes of the whole system.

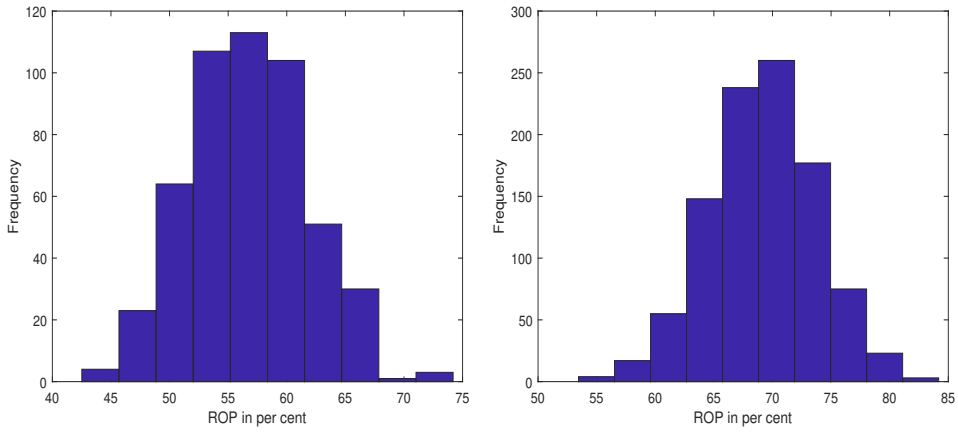


(A) S500 (B) S1000

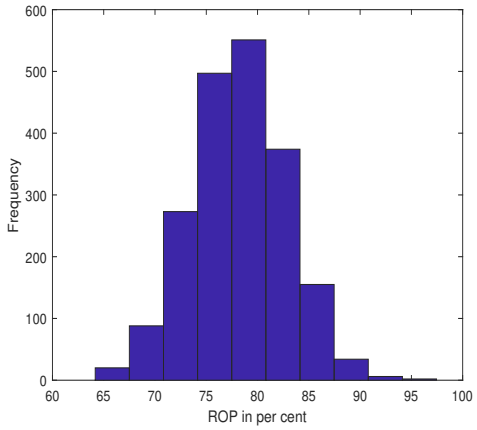


(C) S2000

FIGURE 7.7: The histograms of the average saving amounts that participants (vehicles) experience in the system as a single member



(A) S500 (B) S1000



(C) S2000

FIGURE 7.8: The histograms of the average ROP amounts that participants (vehicles) experience in the system as a single member

Chapter 8

Conclusions and Future Research

In this chapter, the main outcomes of this thesis are discussed and the conclusions are drawn. Following the attempts done in this work, some directions for the future research in the field of fuel or generally cost efficient platooning are recommended. This chapter is arranged as: in Section 8.1, the thesis conclusions are presented, and then, Section 8.2 gives some future research recommendations. Finally, we close the thesis with the summary of this chapter in Section 8.3.

8.1 Outcomes and Conclusions

This thesis focused on the innovative approach of platooning Heavy Duty Vehicles (HDVs) to reduce the total consumed fuel. After discussing the main works done in this field, the importance, novelty and value of the contributions of this thesis are clarified. Two mathematical mixed-integer linear models are presented for the Fuel Efficient Platooning (FEP) problem with real elements like earliest departure times, deadlines, maximum detours and multiple speeds for vehicles. Six different road networks are presented, and a wide rage of samples are built on them. A large volume of experiments done with each of our solution methodologies is an outstanding advantage in this work. The computational hardness of the FEP problem is broken by some changes in the models and pre-assignment of optimal values to a couple of variables.

We could get optimal results by the exact solver of CPLEX for the instances having up to 50 vehicles. It is 5 times larger than the biggest size of a much simpler platooning problem, which has no time restrictions and multiple speeds, solved in *Larsson et al. 2015* [54]. The presented heuristic methods can tackle 10 times larger instances including up to 500 vehicles. This is considerably bigger than the previous FEP problems of this kind solved by any heuristic in the literature (in [54]). A novel heuristic method is proposed called Global Planning (GP), which is able to attain very good results in a much shorter time in comparison to its counterparts.

The solution process of the FEP problem is done in two parts, which is a very practical approach used in the heuristic and meta-heuristic methodologies of this thesis. The first part provides a main structure for solutions by determining vehicles' routing, and the second part is scheduling the travel time and adjusting the speeds of vehicles by the complementary algorithms. Specially, the introduced scheduling and embedded deadline violation resolution algorithm, which give an excellent travel time plan for any routing, are important outcomes. Thanks these algorithms, we can evaluate solutions by calculating their objective value. This two-step approach is specific of our work.

To the best of our knowledge, there have not been any significant attempt in applying meta-heuristics to the platooning problem. However, in this research, three meta-heuristics, namely GA, ACO and PSO, are adapted and employed to deal with large instances of the FEP problem and their outcomes are very satisfactory in terms of solution quality and time. A part of the meta-heuristics' success can be due to the appropriate parameter setting with the RSM method. Although other

parameter optimisation methods can be used alternatively, but their long running time is a limiting factor. When judged by both criteria of saving and solution time together, the ACO method is superior to the GA and PSO. Hence, this ACO can be used as the most efficient tool to solve any FEP instance.

A significant advantage is doing numerous experiments on 6 various road networks. Analysing the results of the networks, it can be concluded that by wider networks the saving decreases. Although complex networks like Grid50 increase the solution time, their effect on the time is much less than increasing the number of vehicles. The instances with up to 2000 vehicles are tackled in this research, however, larger instances can be solved by the ACO and other meta-heuristics if more time is spent than our time limit. The fact is that meta-heuristic schemes can provide fast solutions, however, in our case, the algorithms which must be used for the solution evaluation consume the time and this extremely slows down the optimisation process.

The results verify good savings, which are for the largest instances on average almost 7.5%, which is near to the maximum possible saving that can be achieved by the basic saving factor of platooning ($\eta = 0.1$) considering the time and distance restrictions. Nonetheless, as mentioned and shown in the sensitivity analyses, the saving can be even further grown if the saving factor of platooning (η) also grows through entering new technologies and infrastructures. This is the factor which has the most influence in boosting the platooning benefit. In addition, increasing the number of vehicles raises the saving but the added saving gradually falls as more vehicles exist in the system. The reason is that the most of platooning opportunities have been already used.

Another significant conclusion is that the importance of speed adjustment and routing decreases by larger number of participating vehicles. This is because there are sufficient platooning opportunities with the cruise speed on the shortest path(s) of vehicles to dispense with making detours or accelerating them to catch far platoons. This gives us the idea of excluding these two attempts in solving instances larger than those solved in this research in order to reduce the solution time.

One may think that the achieved fuel saving is low specially in small-sized samples. However, if you consider the large amount of fuel consumption in the real transportation systems, and the huge related costs and emissions, even a small saving percentage means that large expense and environmental pollution are avoided.

Generally, this thesis has deeply researched into FEP problem and proposed several alternative solution methodologies and complementary algorithms to solve it. These tools are very helpful in real cases and can be applied by transportation companies to provide financial benefits and more importantly, environmental protection.

8.2 Future Research Directions

In this section, some recommendations are presented for extending the research into fuel efficient platooning. An important characteristic of any real transportation system is that the parameters and inputs can easily change as time passes. We have partly discussed this aspect in Chapter 7 and the sensitivity analyses are conducted in this direction. However, to deal with the changeability of inputs, a thorough approach could be real time optimisation. There are several methods in this field but due to the high complexity of the FEP problem, applying them is very difficult and time consuming. A valuable future direction can be devising appropriate real time methodologies for solving the problem.

In this thesis, some real elements such as earliest departure times, deadlines, multiple speed options, and maximum allowable detour are considered in the FEP models. Nonetheless, it is strongly recommended to research and find more

elements and features that can exist in real transportation systems and within companies, and then, to build new models that include them.

The fuel efficient platooning can be converted to a general problem called the cost efficient platooning if we can find and realise other cost reduction sources which are achievable through platooning. Connections and reciprocal interactions with transportation companies can help us in this way. As already mentioned, one of them is autonomous driving of the following vehicles of platoons. In this approach, a single driver in the leader vehicle conducts all the vehicles of a platoon. Since the largest share in the transportation cost belongs to human wage according to Figure 1.1 in Chapter 1, this approach can considerably increases the unit cost reduction factor of vehicles by platooning (η), and subsequently, the total saving of the system. So the importance of solving a platooning problems grows considerably and more attention can be caught.

The methodologies proposed by this thesis can be applied to real cases obtained from transportation companies. In the case of instances with larger sizes, a recommendation is attempting the decomposition approaches which can easily break the whole problem into some quickly solvable sub-problems. Another effective approach in dealing with such a platooning problem is using clustering techniques because they can put vehicles in groups to drive together as a platoon. Although they have been partly used in the literature, further works and developments are in demand.

8.3 Summary

In this chapter, we presented the overall conclusions and highlighted some important results of the previous chapters. It was tried to explain the main outcomes from another aspect which is different from those presented in the chapters. Then, some important directions to continue the research into cost efficient platooning were recommended. However, other interesting working areas in vehicle platooning can be found by interested researchers. In the end, it is worth noting that vehicle platooning is a very wide research field, and therefore, needs too much attempt and work in the future to make its complete implementation possible and realise its maximum benefit.

List of Figures

List of Tables

Bibliography

[1] Scania CV AB. *Annual Report, March 2013*. URL: https://www.scania.com/group/en/scania-annual-report-2012-2.

[2] A. Al-Kaisy and C. Durbin. "Platooning on Two-lane Two-way Highways: An Empirical Investigation". In: *Procedia - Social and Behavioral Sciences* 16 (2011). 6th International Symposium on Highway Capacity and Quality of Service, pp. 329 –339. ISSN: 1877-0428. DOI: 10.1016/j.sbspro.2011.04.454. URL: http://www.sciencedirect.com/science/article/pii/S1877042811010007.

[3] A. Alam. "Fuel-Efficient Heavy-Duty Vehicle Platooning". Doctoral thesis. KTH Royal Institute of Technology, School of Electrical Engineering, Stockholm, Sweden, May 2014. URL: https://www.diva-portal.org/smash/get/diva2:719084/FULLTEXT01.pdf.

[4] A. Alam, J. Mårtensson, and K. H Johansson. "Control Engineering Practice Experimental Evaluation of Decentralized Cooperative Cruise Control for Heavy-Duty Vehicle Platooning". In: *Control Engineering Practice* 38 (2015), pp. 11–25. ISSN: 0967-0661. DOI: 10.1016/j.conengprac.2014.12.009. URL: http://dx.doi.org/10.1016/j.conengprac.2014.12.009.

[5] L. Alvarez and R. Horowitz. "Safe Platooning in Automated Highway Systems Part I: Safety Regions Design". In: *Vehicle System Dynamics* 32.1 (1999), pp. 23–55. URL: https://www.tandfonline.com/doi/abs/10.1076/vesd.32.1.23.4228.

[6] L. Alvarez and R. Horowitz. "Safe Platooning in Automated Highway Systems Part II: Velocity Tracking Controller". In: *Vehicle System Dynamics* 32.1 (1999), pp. 57–84. URL: http://www.me.berkeley.edu/~horowitz/Publications_files/Papers_numbered/Journal/Alvarex_VSD_1999_safe_platooning_Part_II.pdf.

[7] C. Bergenhem, E. Hedin, and D. Skarin. "Vehicle-to-Vehicle Communication for a Platooning System". In: *Procedia - Social and Behavioral Sciences* 48 (2012), pp. 1222–1233. ISSN: 1877-0428. DOI: 10.1016/j.sbspro.2012.06.1098. URL: http://dx.doi.org/10.1016/j.sbspro.2012.06.1098.

[8] B. Bergmann and G. Hommel. "Improvements of General Multiple Test Procedures for Redundant Systems of Hypotheses". In: *Multiple Hypothesenprüfung / Multiple Hypotheses Testing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 100–115. ISBN: 978-3-642-52307-6. DOI: 10.1007/978-3-642-52307-6_8. URL: https://link.springer.com/chapter/10.1007/978-3-642-52307-6_8.

[9] A. K. Bhoopalam, Agatz N., and Zuidwijk R. "Planning of Truck Platoons: A Literature Review and Directions for Future Research". In: *Transportation Research Part B: Methodological* 107 (2018), pp. 212 –228. ISSN: 0191-2615. DOI: j.trb.2017.10.016. URL: http://www.sciencedirect.com/science/article/pii/S0191261517305246.

[10] T. Blickle and L. Thiele. "A Comparison of Selection Schemes Used in Evolutionary Algorithms". In: *Evolutionary Computation* 4.4 (1996), pp. 361–394. ISSN: 1063-6560. DOI: 10.1162/evco.1996.4.4.361. URL: https://dl.acm.org/citation.cfm?id=1326732.

[11] J. Bom et al. "A Global Control Strategy for Urban Vehicles Platooning Relying on Nonlinear Decoupling Laws". In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. 2005, pp. 1995–2000. DOI: 10.1109/IROS.2005.1545270. URL: https://ieeexplore.ieee.org/abstract/document/1545270/citations.

[12] C. Bonnet and H. Fritz. "Fuel Consumption Reduction in a Platoon: Experimental Results with two Electronically Coupled Trucks at Close Spacing." In: *Intelligent Vehicle Technology - SP-1558*. Ed. by editor. DOI: 10.4271/2000-01-3056. URL: https://www.sae.org/publications/technical-papers/content/2000-01-3056/.

[13] G. E. P. Box and N. R. Draper. *Response Surfaces, Mixtures, and Ridge Analyses, 2nd Edition*. Wiley-Interscience, 2007. ISBN: 978-0-470-05357-7.

[14] N. Boysen, D. Briskorn, and S. Schwerdfeger. "The Identical-Path Truck Platooning Problem". In: *Transportation Research Part B: Methodological* 109 (2018), pp. 26 –39. ISSN: 0191-2615. DOI: 10.1016/j.trb.2018.01.006. URL: http://www.sciencedirect.com/science/article/pii/S0191261517305970.

[15] B. Calvo and G. Santafe. *SCMAMP: Statistical Comparison of Multiple Algorithms in Multiple Problems*. Oct 2016. URL: https://cran.r-project.org/web/packages/scmamp/scmamp.pdf.

[16] European Commission. *White Paper on Transport - Roadmap to a Single European Transport Area - Towards a Competitive and Resource Efficient Transport System*. European Strategies, Publications Office of the European Union, Luxembourg. URL: https://ec.europa.eu/transport/sites/transport/files/themes/strategies/doc/2011_white_paper/white-paper-illustrated-brochure_en.pdf.

[17] T. H. Cormen et al. *"Section 24.3: Dijkstra's algorithm". Introduction to Algorithms (Second edition).* MIT Press and McGraw–Hill, 2001, 595–601. ISBN: 0-262-03293-7.

[18] GAMS Development Corporation. "General Algebraic Modeling System (GAMS) Release 24.2.1". In: (2013). URL: http://www.gams.com/.

[19] B. Dafflon et al. "Vehicle Platoon and Obstacle Avoidance: A Reactive Agent Approach". In: *IET Intelligent Transport Systems* 7.3 (2013), pp. 257–264. ISSN: 1751-956X. DOI: 10.1049/iet-its.2011.0125.

[20] A. Davila and M. Nombela. "Platooning-Safe and Eco-Friendly Mobility". In: *SAE Technical Papers* (2012). DOI: 10.4271/2012-01-0488. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84877189242&partnerID=40&md5=c25caaba8dd52d8a3460921be427fdc5.

[21] Q. Deng and Ma X. "A Fast Algorithm for Planning Optimal Platoon Speeds on Highway". In: *IFAC Proceedings Volumes*. Vol. 47. 3. 2014, pp. 8073 –8078. DOI: 10.3182/20140824-6-ZA-1003.02778. URL: http://www.sciencedirect.com/science/article/pii/S1474667016428867.

[22] J. Derrac et al. "A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms". In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 3 –18. ISSN: 2210-6502. DOI: 10.1016/j.swevo.2011.02.002. URL: http://www.sciencedirect.com/science/article/pii/S2210650211000034.

[23] M. Di Bernardo et al. "Design, Analysis, and Experimental Validation of a Distributed Protocol for Platooning in the Presence of Time-Varying Heterogeneous Delays". In: *IEEE Transactions on Control Systems Technology* 24.2 (2016), pp. 413–427. DOI: 10.1109/TCST.2015.2437336. URL: https://ieeexplore.ieee.org/document/7134769/.

[24] M. Dorigo. "Optimization, Learning and Natural Algorithms". PhD thesis. Politecnico di Milano, Italy, 1992. ISBN: 9780471671756.

[25] Sungu H. E., Inoue M., and Imura J. i. "Nonlinear spacing policy based vehicle platoon control for local string stability and global traffic flow stability". In: *2015 European Control Conference (ECC)*. 2015, pp. 3396–3401. DOI: 10.1109/ECC.2015.7331059. URL: https://ieeexplore.ieee.org/document/7331059.

[26] M. El Zaher et al. "An Interaction Model for a Local Approach to Vehicle Platoons". In: *International Journal of Vehicle Autonomous Systems* 13 (2016), p. 91. DOI: 10.1504/IJVAS.2016.078760. URL: https://www.inderscienceonline.com/doi/abs/10.1504/IJVAS.2016.078760.

[27] F. Espinosa et al. "Reduction of Lateral and Longitudinal Oscillations of Vehicle's Platooning by Means of Decentralized Overlapping Control". In: *Proceedings of the IEEE Conference on Decision and Control*. 2007, pp. 690–695. DOI: 10.1109/CDC.2007.4434073. URL: https://ieeexplore.ieee.org/document/4434073/.

[28] C.T. Featherstone and M.V. Lowson. "Safety of Automated Passenger Vehicle Platooning". In: vol. 3. 2009, pp. 1665–1673.

[29] P. Fernandes and U. Nunes. "Platooning of Autonomous Vehicles with Intervehicle Communications in SUMO Traffic Simulator". In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. 2010, pp. 1313–1318. DOI: 10.1109/ITSC.2010.5625277. URL: https://ieeexplore.ieee.org/document/5625277/.

[30] P. Fernandes and U. Nunes. "Platooning with IVC-Enabled Autonomous Vehicles: Strategies to Mitigate Communication Delays, Improve Safety and Traffic Flow". In: *IEEE Transactions on Intelligent Transportation Systems* 13.1 (2012), pp. 91–106. DOI: 10.1109/TITS.2011.2179936. URL: https://ieeexplore.ieee.org/document/6135795/.

[31] "GAMS - The Solver Manuals, GAMS Release 24.2.1". In: (2013). URL: http://www.gams.com/dd/docs/solvers/allsolvers.pdf.

[32] S. Gao, A. Lim, and D. Bevly. "An empirical study of DSRC V2V performance in truck platooning scenarios". In: *Digital Communications and Networks* 2.40 (2016). Next Generation Wireless Communication Technologies, pp. 233 –244. ISSN: 2352-8648. DOI: 10.1016/j.dcan.2016.10.003. URL: http://www.sciencedirect.com/science/article/pii/S235286481630075X.

[33] J.L. Gattis et al. "Rural Two-Lane Passing Headways and Platooning". In: *Transportation Research Record* 1579 (1997), pp. 27–34. DOI: 10.3141/1579-04. URL: https://trrjournalonline.trb.org/doi/abs/10.3141/1579-04.

[34] M. Goli and A. Eskandarian. "Evaluation of Lateral Trajectories with Different Controllers for Multi-Vehicle Merging in Platoon". In: 2014, pp. 673–678. DOI: 10.1109/ICCVE.2014.7297633. URL: https://ieeexplore.ieee.org/document/7297633/.

[35] C. Guo et al. "Self-Defensive Coordinated Maneuvering of an Intelligent Vehicle Platoon in Mixed Traffic". In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. 2012, pp. 1726–1733. DOI: 10.1109/ITSC.2012.6338836. URL: https://ieeexplore.ieee.org/document/6338836/.

[36] *Gurobi in GAMS*. URL: https://www.gams.com/latest/docs/S_GUROBI.html.

[37] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms*. Wiley Inter-Science electronic collection. Wiley, 2004. ISBN: 9780471671756. URL: https://books.google.de/books?id=5gR97w4X0EkC.

[38] *Highway Capacity Manual*. Transportation Research Board, National Research Council, 2012. URL: http://hcm.trb.org.

[39] L.-N. Hoang, E. Uhlemann, and M. Jonsson. "An Efficient Message Dissemination Technique in Platooning Applications". In: *IEEE Communications Letters* 19.6 (2015), pp. 1017–1020. DOI: 10.1109/LCOMM.2015.2416174. URL: https://ieeexplore.ieee.org/abstract/document/7066951/authors.

[40] S. Van de Hoef, K. H. Johansson, and D. V. Dimarogonas. "Computing Feasible Vehicle Platooning Opportunities for Transport Assignments". In: *IFAC-PapersOnLine* 49.3 (2016), pp. 43–48. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2016.07.008". URL: http://www.sciencedirect.com/science/article/pii/S2405896316302038.

[41] S.V.D. Hoef, K.H. Johansson, and D.V. Dimarogonas. "Coordinating Truck Platooning by Clustering Pairwise Fuel-Optimal Plans". In: vol. 2015-October. 2015, pp. 408–415. DOI: 10.1109/ITSC.2015.75. URL: https://ieeexplore.ieee.org/document/7313167/.

[42] A. Kaku, M. Masakazu, and T. Kawabe. "A Centralized Control System for Ecological Vehicle Platooning Using Linear Quadratic Regulator Theory". In: *Artificial Life and Robotics* 17.1 (2012), pp. 70–740. ISSN: 1614-7456. DOI: 10.1007/s10015-012-0019-3. URL: https://link.springer.com/article/10.1007/s10015-012-0019-3.

[43] C. Kammer. "Coordinated Heavy Truck Platoon Routing Using Global and Locally Distributed Approaches". Master thesis. KTH Royal Institute of Technology, School of Electrical Engineering, Stockholm, Sweden, April 2013. URL: http://www.diva-portal.org/smash/get/diva2:855061/FULLTEXT01.pdf.

[44] K. Karlsson et al. "Evaluation of the V2V channel and diversity potential for platooning trucks". In: *2016 10th European Conference on Antennas and Propagation, EuCAP 2016*. 2016. DOI: 10.1109/EuCAP.2016.7481942. URL: https://ieeexplore.ieee.org/abstract/document/7481942/.

[45] R. M. Karp. "Reducibility Among Combinatorial Problems". In: *Complexity of Computer Computations*. Ed. by R. E. Miller and J. W. Thatcher. Plenum Press, 1972, pp. 85–103.

[46] P. Kavathekar and Y. Chen. "Vehicle Platooning: A Brief Survey and Categorization". In: *Proceedings of the ASME Design Engineering Technical Conference*. Vol. 3. 2011, pp. 829–845. DOI: 10.1115/DETC2011-47861. URL: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1640080.

[47] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Neural Networks, 1995. Proceedings., IEEE International Conference on*. Vol. 4. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[48] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001. ISBN: 1-55860-595-9.

[49] W. Kühn, M. Müller, and T. Höppner. "Road Data as Prior Knowledge for Highly Automated Driving". In: *Transportation Research Procedia* 27 (2017). 20th EURO Working Group on Transportation Meeting, EWGT 2017, 4-6 September 2017, Budapest, Hungary, pp. 222 –229. ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2017.12.011. URL: http://www.sciencedirect.com/science/article/pii/S2352146517309080.

[50] R. Kianfar, P. Falcone, and J. Fredriksson. "A Control Matching Model Predictive Control Approach to String Stable Vehicle Platooning". In: *Control Engineering Practice* 45 (2015), pp. 163–173. ISSN: 0967-0661. DOI: 10.1016/j.conengprac.2015.09.011. URL: http://dx.doi.org/10.1016/j.conengprac.2015.09.011.

[51] J.P.J. Koller et al. "Fuel-Efficient Control of Merging Maneuvers for Heavy-Duty Vehicle Platooning". In: vol. 2015-October. 2015, pp. 1702–1707. DOI: 10.1109/ITSC.2015.276. URL: https://ieeexplore.ieee.org/document/7313368/.

[52] J. Larson, T. Munson, and V. Sokolov. "Coordinated Platoon Routing in a Metropolitan Network". In: *2016 Proceedings of the Seventh SIAM Workshop on Combinatorial Scientific Computing*, pp. 73–82. DOI: 10.1137/1.9781611974690.ch8. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611974690.ch8.

[53] J. Larson et al. "Coordinated Route Optimization for Heavy-Duty Vehicle Platoons". In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. 2013, pp. 1196–1202. DOI: 10.1109/ITSC.2013.6728395. URL: https://ieeexplore.ieee.org/document/6728395/.

[54] E. Larsson, G. Sennton, and J. Larson. "The Vehicle Platooning Problem : Computational Complexity and Heuristics". In: *Transportation Research Part C* 60 (2015), pp. 258–277. ISSN: 0968-090X. DOI: 10.1016/j.trc.2015.08.019. URL: https://www.sciencedirect.com/science/article/pii/S0968090X15003204.

[55] M. Larsson et al. "A Data Age Dependent Broadcast Forwarding Algorithm for Reliable Platooning Applications". In: *Mobile Information Systems* 2016 (2016). DOI: 10.1155/2016/7489873. URL: https://www.hindawi.com/journals/misy/2016/7489873/.

[56] W. Levine and M. Athans. "On the Optimal Error Regulation of a String of Moving Vehicles". In: *IEEE Transactions on Automatic Control* 11.3 (1966), pp. 355–361. ISSN: 0018-9286. DOI: 10.1109/TAC.1966.1098376. URL: https://ieeexplore.ieee.org/document/1098376/.

[57] B. Li. "Stochastic Modeling for Vehicle Platoons (I): Dynamic Grouping Behavior and Online Platoon Recognition". In: *Transportation Research Part B: Methodological* 95 (2017), pp. 364 –377. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2016.07.019. URL: http://www.sciencedirect.com/science/article/pii/S0191261515301259.

[58] B. Li. "Stochastic Modeling for Vehicle Platoons (II): Statistical Characteristics". In: *Transportation Research Part B: Methodological* 95 (2017), pp. 378 –393. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2016.07.017. URL: http://www.sciencedirect.com/science/article/pii/S0191261515301260.

[59] K.-Y. Liang. "Coordination and Routing for Fuel-Efficient Heavy-Duty Vehicle Platoon Formation". Licentiate thesis. KTH Royal Institute of Technology, School of Electrical Engineering, Stockholm,

Sweden, March 2014. URL: https://www.diva-portal.org/smash/get/diva2:706818/FULLTEXT01.pdf.

[60] K.-Y. Liang et al. "The Influence of Traffic on Heavy-Duty Vehicle Platoon Formation". In: *IEEE Intelligent Vehicles Symposium, Proceedings*. Vol. 2015-August. 2015, pp. 150–155. DOI: 10.1109/IVS.2015.7225678. URL: https://ieeexplore.ieee.org/document/7225678/.

[61] S. Linsenmayer and D. V. Dimarogonas. "Event-Triggered Control for Vehicle Platooning". In: *2015 American Control Conference (ACC)*. 2015, pp. 3101–3106. DOI: 10.1109/ACC.2015.7171809. URL: https://ieeexplore.ieee.org/document/7171809.

[62] J. Lioris et al. "Doubling throughput in urban roads by platooning". In: *IFAC-PapersOnLine* 49.3 (2016). 14th IFAC Symposium on Control in Transportation SystemsCTS 2016, pp. 49–54. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2016.07.009. URL: http://www.sciencedirect.com/science/article/pii/S240589631630204X.

[63] X.-Y. Lu and K.J. Hedrick. "Longitudinal Control Algorithm for Automated Vehicle Merging". In: *Proceedings of the IEEE Conference on Decision and Control*. Vol. 1. 2000, pp. 450–455. DOI: 10.1109/CDC.2000.912805. URL: https://ieeexplore.ieee.org/document/912805/.

[64] X.-Y. Lu and S. Shladover. "Automated Truck Platoon Control and Field Test, Road Vehicle Automation". In: *Road Vehicle Automation, August 2014*. Springer International Publishing. ISBN: 978-3-319-05990-7. DOI: https://doi.org/10.1007/978-3-319-05990-7_21. URL: https://link.springer.com/chapter/10.1007/978-3-319-05990-7_21.

[65] F. Luo, J. Larson, and T. Munson. "Coordinated Platooning with Multiple Speeds". In: *Transportation Research Part C: Emerging Technologies* 90 (2018), pp. 213 –225. ISSN: 0968-090X. DOI: 10.1016/j.trc.2018.02.011. URL: http://www.sciencedirect.com/science/article/pii/S0968090X18302079.

[66] J.P. Masehuw, G.C. Keßler, and D. Abel. "Control Concepts for Safe Vehicle Platooning". In: *FISITA World Automotive Cong-ress 2008, Congress Proceedings - Mobility Concepts, Man Machine Interface, Process Challenges, Virtual Reality*. Vol. 1. 2008, pp. 74–83.

[67] F. Michaud et al. "Coordinated Maneuvering of Automated Vehicles in Platoons". In: *IEEE Transactions on Intelligent Transportation Systems* 7.4 (2006), pp. 437–446. DOI: 10.1109/TITS.2006.883939. URL: https://ieeexplore.ieee.org/document/4019445/.

[68] A. Nourmohammadzadeh and S. Hartmann. "Fuel-efficient truck platooning by a novel meta-heuristic inspired from ant colony optimisation". In: *Soft Computing* (2018). ISSN: 1433-7479. DOI: 10.1007/s00500-018-3518-x. URL: https://doi.org/10.1007/s00500-018-3518-x.

[69] A. Nourmohammadzadeh and S. Hartmann. "Fuel Efficient Truck Platooning with Time Restrictions and Multiple Speeds Solved by a Particle Swarm Optimisation". In: *Theory and Practice of Natural Computing*. Cham: Springer International Publishing, 2018, pp. 188–200. ISBN: 978-3-030-04070-3. DOI: 10.1007/978-3-030-04070-3_15. URL: https://link.springer.com/chapter/10.1007/978-3-030-04070-3_15.

[70] A. Nourmohammadzadeh and S. Hartmann. "The Fuel-Efficient Platooning of Heavy Duty Vehicles by Mathematical Programming and Genetic Algorithm". In: *Theory and Practice of Natural Computing*. Cham: Springer International Publishing, 2016, pp. 46–57. ISBN: 978-3-319-49001-4. DOI: 10.1007/978-3-319-49001-4_4. URL: https://link.springer.com/chapter/10.1007/978-3-319-49001-4_4.

[71] C. Nowakowski et al. "Cooperative Adaptive Cruise Control: Driver Acceptance of Following Gap Settings Less than One Second". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 54.24 (2010), pp. 2033–2037. DOI: 10.1177/154193121005402403. URL: https://doi.org/10.1177/154193121005402403.

[72] T. Ogitsu et al. "Decision Process for Handling Operations Against Device Failures in Heavy Duty Trucks in a Platoon". In: *9th FORMS/FORMAT 2012 - Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems- Braunschweig, Germany*. 2012, pp. 71–80. URL: https://keio.pure.elsevier.com/ja/publications/decision-process-for-handling-operat-ions-against-device-failures-.

[73] P. Paoletti and G. Innocenti. "Effects of Nearest Neighbors Interactions on Control of Nonlinear Vehicular Platooning". In: *2015 European Control Conference, ECC 2015*. 2015, pp. 2991–2996. DOI: 10.1109/ECC.2015.7330992. URL: https://ieeexplore.ieee.org/document/7330992/.

[74] M. Parent, P. Daviet, and S. Abdou. "Vision Technique for Platoon Driving". In: *Proceedings of the International Conference on Applications of Advanced Technologies in Transportation Engineering*. 1996, pp. 666–675.

[75] R Development Core Team. *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria, 2008. URL: http://www.R-project.org.

[76] K. R. Ranjit. *Design of Experiments Using The Taguchi Approach: 16 Steps to Product and Process Improvement*. Wiley-Interscience, 2001. ISBN: 978-0-471-36101-5.

[77] T. Robinson, E. Chan, and E. Coelingh. "Operating platoons on public motorways: an introduction to the SARTRE platooning programme." In: *Proceedings of the 17th ITS World Congress, June 2010*. Ed. by editor. URL: https://www.researchgate.net/publication/268300380_Operating_Platoons_On_Public_Motorways_An_Introduction_To_The_SARTRE_Platooning_Programme.

[78] M. Schittler. "State-of-the-Art and Emerging Truck Engine Technologies for Optimized Performance, Emissions and Life Cycle Costs". In: *9th Diesel Emissions Reduction Conference, August 2003, Rhode Island, USA, August 2003*. URL: http://osti.gov/scitech/servlets/purl/829810.

[79] D. Schrank, B. Eisele, and T. Lomax. *2012 Urban Mobility Report Powered by INRIX Traffic Data*. type. Sponsored by: Southwest Region University Transportation Center (SWUTC), Texas A&M Transportation Institute, The Texas A&M University System, July 2015. URL: https://static.tti.tamu.edu/swutc.tamu.edu/publications/technicalreports/161302-1.pdf.

[80] A. Schroten, G. Warringa, and M. Bles. "Marginal Abatement Cost Curves for Heavy Duty Vehicles". In: *Background report. CE Delft, Delft, Netherlands, July 2012*. URL: https://ec.europa.eu/clima/sites/clima/files/transport/vehicles/heavy/docs/hdv_2012_co2_abatement_cost_curves_en.pdf.

[81] M. Segata et al. "Toward Communication Strategies for Platooning: Simulative and Experimental Evaluation". In: *IEEE Transactions on Vehicular Technology* 64.12 (2015), pp. 5411–5423. DOI: 10.1109/TVT.2015.2489459. URL: https://ieeexplore.ieee.org/document/7295637.

[82] Y. Shi and R. Eberhart. "A Modified Particle Swarm Optimizer". In: *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*. 1998, pp. 69–73. DOI: 10.1109/ICEC.1998.699146. URL: https://ieeexplore.ieee.org/document/699146.

[83] S. E. Shladover. *Recent International Activity in Cooperative Vehicle–Highway Automation Systems*. University of California, Berkeley, 2012. URL: https://trid.trb.org/view/1238491.

[84] Inc. State College PA: Minitab. URL: https://www.minitab.com.

[85] D. Steinmetz, G. Burmester, and S. Hartmann. "A Fast Heuristic for Finding Near-Optimal Groups for Vehicle Platooning in Road Networks". In: *Database and Expert Systems Applications*. Cham: Springer International Publishing, 2017, pp. 395–405. DOI: 10.1007/978-3-319-64471-4_32. URL: https://link.springer.com/chapter/10.1007/978-3-319-64471-4_32.

[86] K. Tadakuma et al. "Prediction Formula of Aerodynamic Drag Reduction in Multiple-Vehicle Platooning Based on Wake Analysis and On-Road Experiments". In: *SAE International Journal of Passenger Cars - Mechanical Systems* 9.2 (2016). DOI: 10.4271/2016-01-1596. URL: https://www.sae.org/publications/technical-papers/content/2016-01-1596/.

[87] *The 1939 New York World's Fair*. General Motors, 1939. URL: http://www.archive.org/details/ToNewHor1940.

[88] *The World Business Council for Sustainable Development (WBCSD)*. URL: https://www.wbcsd.org.

[89] V. Turri, B. Besselink, and K.H. Johansson. "Cooperative Look-Ahead Control for Fuel-Efficient and Safe Heavy-Duty Vehicle Platooning". In: *IEEE Transactions on Control Systems Technology* (2016). DOI: 10.1109/TCST.2016.2542044. URL: https://ieeexplore.ieee.org/document/7445860/.

[90] P. Varaiya. "Smart Cars on Smart Roads: Problems of Control". In: *IEEE Transactions on Automatic Control* 38.2 (1993), pp. 195–207. ISSN: 0018-9286. DOI: 10.1109/9.250509. URL: https://ieeexplore.ieee.org/document/250509/.

[91] D. Wang, M. Pham, and C. T. Phampt. "Simulation Study of Vehicle Platooning Maneuvers with Full-State Tracking Control". In: *Modeling, Simulation and Optimization of Complex Processes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 539–548. ISBN: 978-3-540-27170-3.

[92] *Weltweites Frachtvolumen im Vergleich der Jahre 2010 und 2050 nach Verkehrsträgern (in Billionen Tonnenkilometer)*. URL: https://de.statista.com/statistik/daten/studie/482955/umfrage/frachtvolumen-weltweit-nach-verkehrstraegern/.

[93] W. van Willigen, E. Haasdijk, and L. Kester. "A Multi-objective Approach to Evolving Platooning Strategies in Intelligent Transportation Systems". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. GECCO '13. Amsterdam, The Netherlands: ACM, 2013, pp. 1397–1404. ISBN: 978-1-4503-1963-8. DOI: 10.1145/2463372.2463534. URL: http://doi.acm.org/10.1145/2463372.2463534.

[94] C. C. de Wit and B. Brogliato. "Stability Issues for Vehicle Platooning in Automated Highway Systems". In: vol. 2. 1999, pp. 1377–1382. DOI: 10.1109/CCA.1999.801173. URL: https://ieeexplore.ieee.org/document/801173/.

[95] H. Wolf-Heinrich and S. R. Ahmed. "Aerodynamics of Road Vehicles." In: *Warrendale: Society of Automotive Engineers* (1998).

[96] L. Xu et al. "Communication Information Structures and Contents for Enhanced Safety of Highway Vehicle Platoons". In: *IEEE Transactions on Vehicular Technology* 63.9 (2014), pp. 4206–4220. DOI: 10.1109/TVT.2014.2311384. URL: https://ieeexplore.ieee.org/document/6766275/.

[97] L. Xu et al. "Coordinated Control and Communication for Enhanced Safety of Highway Vehicle Platoons". In: *2013 International Conference on Connected Vehicles and Expo, ICCVE 2013 - Proceedings*. 2013, pp. 43–47. DOI: 10.1109/ICCVE.2013.6799767. URL: https://ieeexplore.ieee.org/document/6799767/.

[98] K. Yu et al. "Model Predictive Control for Hybrid Electric Vehicle Platooning Using Route Information". In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 230.9 (2016), pp. 1273–1285. DOI: 10.1177/0954407015606314. URL: https://journals.sagepub.com/doi/pdf/10.1177/0954407015606314.

[99] K. Yu et al. "Model Predictive Control for Hybrid Electric Vehicle Platooning Using Slope Information". In: *IEEE Transactions on Intelligent Transportation Systems* 17.7 (2016), pp. 1894–1909. DOI: 10.1109/TITS.2015.2513766. URL: https://ieeexplore.ieee.org/document/7390284.

[100] W. Zhang, M. Sundberg, and A. Karlström. "Platoon Coordination with Time Windows: An Operational Perspective". In: *Transportation Research Procedia* 27 (2017). 20th EURO Working Group on Transportation Meeting, EWGT 2017, 4-6 September 2017, Budapest, Hungary, pp. 357 –364. ISSN: 2352-1465. DOI: 10.1016/j.trpro.2017.12.129. URL: http://www.sciencedirect.com/science/article/pii/S2352146517310268.

[101] S. Zhao et al. "Vehicle to Vehicle Communication and Platooning for EV with Wireless Sensor Network". In: *2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan, SICE 2015*. 2015, pp. 1435–1440. DOI: 10.1109/SICE.2015.7285493. URL: https://ieeexplore.ieee.org/document/7285493.

[102] Y. Zheng et al. "Stability Margin Improvement of Vehicular Platoon Considering Undirected Topology and Asymmetric Control". In: *IEEE Transactions on Control Systems Technology* 24.4 (2016), pp. 1253–1265. DOI: 10.1109/TCST.2015.2483564. URL: https://ieeexplore.ieee.org/document/7299636/.